

[illegible][illegible]

PPPPPPPP		AAAAAA	EEEEEEEEEE	RRRRRRRR	RRRRRRRR	000000	RRRRRRRR
PPPPPPPP		AAAAAA	EEEEEEEEEE	RRRRRRRR	RRRRRRRR	000000	RRRRRRRR
PP	PP	AA	EE	RR	RR	00	RR
PP	PP	AA	EE	RR	RR	00	RR
PP	PP	AA	EE	RR	RR	00	RR
PPPPPPPP		AA	EEEEEEEE	RRRRRRRR	RRRRRRRR	00	RRRRRRRR
PPPPPPPP		AA	EEEEEEEE	RRRRRRRR	RRRRRRRR	00	RRRRRRRR
PP		AAAAAAAAAA	EE	RR	RR	00	RR
PP		AAAAAAAAAA	EE	RR	RR	00	RR
PP		AA	EE	RR	RR	00	RR
PP		AA	EE	RR	RR	00	RR
PP		AA	EEEEEEEEEE	RR	RR	000000	RR
PP		AA	EEEEEEEEEE	RR	RR	000000	RR

....  
....  
....  
....

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SSSSSS
LL	II	SSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS

(2)	380	DEFINITIONS	
(3)	422	OPAO ERROR LOGGING DATA	
(10)	629	ERR\$CRASHVC,	CRASH VC ON SPECIFIED
(10)	630	-	PATH BLOCK
(11)	690	ERR\$CRASH_PORT,	INIT PORT CRASH
(12)	750	ERR\$PWF_RECOV,	NOTIFY SYSAPS WITH
(12)	751	-	CONNECTIONS ON POWER
(12)	752	-	FAILED PORT
(13)	976	UNLOCK BADQ,	ZERO CORRUPTED QUEUE HDRS
(14)	1013	ERR\$DISC_PWFAIL,	PROCESS DISCONNECT CALL
(14)	1014	-	FOR CDT ON POWER
(14)	1015	-	FAILED PORT
(15)	1079	ERR\$CLEANUP_PKT	CLEAN UP PACKETS QUEUED TO
(15)	1080	-	PORT AND IN LOGOUT AREA
(16)	1137	FLUSH_Q	REMOVE AND DISPOSE OF
(16)	1138	-	ALL QUEUED ENTRIES
(16)	1139	ERR\$DISP_ENTRY	DISPOSE OF A SINGLE ENTRY
(17)	1201	ERR\$INIPORT,	CALL PORT HARDWARE INIT
(18)	1240	ERR\$BUGCHECK,	RECORD PORT LOCAL STORE
(18)	1241	-	IN MEMORY
(18)	1242	ERR\$BUGCHECKNF,	RECORD LOCAL STORE CONDITIONALLY
(18)	1243	-	IF NONFATAL BUGCHECKS ARE FATAL
(19)	1288	ERR\$DEBUGCHECK,	DEBUG BUGCHECK ENABLE FLAGS
(20)	1392	ELOG\$INIT_SWERR,	LOG SOFTWARE ERROR
(20)	1393	-	ENCOUNTERED DURING
(20)	1394	-	PORT INITIALIZATION
(20)	1395	ELOG\$UCODE_NORD,	LOG MICROCODE NOT
(20)	1396	-	PROPERLY READ BACK
(20)	1397	-	ERROR
(20)	1398	ELOG\$HARDWARE,	LOG HARDWARE ERROR
(20)	1399	ELOG\$Q_INTRLOCK,	LOG QUEUE INTERLOCK
(20)	1400	-	FAILURE
(21)	1615	ELOG\$REGDUMP,	DEVICE ATTENTION
(21)	1616	-	REGISTER DUMP ROUTINE
(22)	1665	ELOG\$PACKET,	LOG PACKET RELATED
(22)	1666	-	ERROR, GENERAL CASE
(22)	1667	ELOG\$CABLES,	LOG CABLE STATUS
(22)	1668	-	CHNAGE, GENERAL CASE
(22)	1669	ELOG\$PTH_ST_CHG	LOG PATH STATUS
(22)	1670	-	CHANGE
(22)	1671	ELOG\$CBL_X_CHG	LOG CABLES CROSSED OR
(22)	1672	-	NOT CROSSED STATUS
(22)	1673	-	CHANGE
(22)	1674	ELOG\$ERROR_DG	LOG ERROR LOG DATAGRAM
(24)	1974	OPAO_LOG,	OPAO ERROR LOGGING ROUTINE
(29)	2105	OPAO_LOG_FORK,	OPAO ERROR LOGGING
(29)	2106	-	FORK PROCESS ROUTINE
(30)	2184	OPAO ERROR LOGGING FORMATTING ROUTINES	
(30)	2185	ERR\$CNV_HEX_DEC	ROUTINE TO CONVERT A BINARY NUMBER
(30)	2186	-	INTO ITS DECIMAL ASCII EQUIVALENCE
(31)	2224	FORMAT_PKT,	ROUTINE TO FORMAT PACKET
(31)	2225	-	INFORMATION
(32)	2270	FORMAT_PORT,	ROUTINE TO FORMAT A
(32)	2271	-	REMOTE PORT NUMBER
(33)	2304	FORMAT_REGS,	ROUTINE TO FORMAT PORT
(33)	2305	-	REGISTERS
(34)	2352	FORMAT_REV,	FORMAT PORT UCODE REV LEVELS
(35)	2398	HEX_TO_ASCII	ROUTINE TO CONVERT A BINARY NUMBER
(35)	2399	-	INTO ITS ASCII EQUIVALENCE



```
0000 1 .TITLE PAERROR Error Handling & Logging Routines
0000 2 .IDENT 'V04-001'
0000 3
0000 4 *****
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9 *
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *
0000 25 *****
0000 26
0000 27 ++
0000 28
0000 29 FACILITY:
0000 30
0000 31 VAX/VMS EXECUTIVE, I/O DRIVERS
0000 32
0000 33 ABSTRACT: ROUTINES TO HANDLE CI VIRTUAL CIRCUIT RECOVERY
0000 34
0000 35 AUTHOR: N. KRONENBERG, DECEMBER 1981
0000 36
0000 37 MODIFIED BY:
0000 38
0000 39 V04-001 NPK3066 N. Kronenberg 7-Sep-1984
0000 40 Disable invalid buffer name bugcheck since bug is found.
0000 41 With this edit, all optional bugchecks are disabled
0000 42 and maximum error recovery enabled.
0000 43
0000 44 V03-040 NPK3065 N. Kronenberg 23-Aug-1984
0000 45 Disable MFQE optional bugcheck since bug is found.
0000 46
0000 47 V03-039 NPK3061 N. Kronenberg 9-Aug-1984
0000 48 Remove optional debug bugcheck on unexpected port
0000 49 interrupt.
0000 50
0000 51 V03-038 NPK3060 N. Kronenberg 1-Aug-1984
0000 52 Remove remote port from OPA0 messages concerning
0000 53 loopback dgs since no remote port is applicable.
0000 54 Make loopback dg OPA0 messages be reported always.
0000 55
0000 56 V03-037 NPK3058 N. Kronenberg 25-Jul-1984
0000 57 Add MFQE optional bugcheck enable flag and enable
```

0000 58 : three kinds of optional bugchecks.  
0000 59 :  
0000 60 : V03-036 NPK3057 N. Kronenberg 23-Jul-1984  
0000 61 : Change the OPA0 message descriptors for cpu/port  
0000 62 : ucode not at required rev level not to include  
0000 63 : offline messages since these are generated separately  
0000 64 : in PAINIT, CLEANUP\_PDT.  
0000 65 :  
0000 66 : V03-035 NPK3055 N. Kronenberg 14-Jul-1984  
0000 67 : Change OPA0 error log msgs for cpu/port ucode rev  
0000 68 : error to include port offline msg. Change wording  
0000 69 : of cpu ucode rev error msg to say that rev is insufficient  
0000 70 : for CI activity.  
0000 71 : Add separate port ucode rev warning msg that does not  
0000 72 : include offline announcement.  
0000 73 : Add ELOG\$CPU\_REV, ELOG\$UCODE\_ERR, ELOG\_UCODE\_WARN.  
0000 74 :  
0000 75 : V03-034 NPK3054 N. Kronenberg 24-Jun-1984  
0000 76 : Add OPA0 messages to warn operator of either CPU  
0000 77 : rev level insufficient to support ci, or the ci  
0000 78 : ucode rev level is insufficient.  
0000 79 :  
0000 80 : V03-033 NPK3053 N. Kronenberg 17-May-1984  
0000 81 : Fix branch error in NPK3052.  
0000 82 :  
0000 83 : V03-032 NPK3052 N. Kronenberg 4-May-1984  
0000 84 : Fix ERR\$PWF\_RECOV to properly handle a port failure  
0000 85 : for a port with circuits in VC\_FAIL state.  
0000 86 :  
0000 87 : V03-031 NPK3048 N. Kronenberg 9-Apr-1984  
0000 88 : Add two new \$DEBUGCHECK enable flags.  
0000 89 :  
0000 90 : V03-030 TMK0005 Todd M. Katz 25-Mar-1984  
0000 91 : Change the text of the remote system conflicts \_OPA0 error  
0000 92 : logging message.  
0000 93 :  
0000 94 : V03-029 TMK0004 Todd M. Katz 24-Mar-1984  
0000 95 : When it is decided to log an error condition to \_OPA0, a fork  
0000 96 : process is created to format and broadcast an appropriate  
0000 97 : message. It is absolutely necessary that all messages be  
0000 98 : formatted at fork IPL. This is because there is only one copy  
0000 99 : of each message, but there maybe multiple CI ports making use  
0000 100 : of each message.  
0000 101 :  
0000 102 : However, what is incorrect is that the optional data which maybe  
0000 103 : used for formatting a \_OPA0 error log message is being extracted  
0000 104 : from the UCB error logging buffer or from the device registers  
0000 105 : within the context of the fork process. By the time the fork  
0000 106 : process gets a chance to execute and make use of this optional  
0000 107 : data for formatting a message, it is possible (and in the case  
0000 108 : of device registers certain) that the values stored in these  
0000 109 : locations will have changed.  
0000 110 :  
0000 111 : The solution to this problem is to store the needed information  
0000 112 : within UCB\$T\_OPA0\_TEMP (a new UCB field three longwards in size)  
0000 113 : just before the creation of the fork process within OPA0\_LOG.  
0000 114 : Then, whenever optional formatting of an \_OPA0 error log message



0000 115 :  
0000 116 :  
0000 117 :  
0000 118 :  
0000 119 :  
0000 120 :  
0000 121 :  
0000 122 :  
0000 123 :  
0000 124 :  
0000 125 :  
0000 126 :  
0000 127 :  
0000 128 :  
0000 129 :  
0000 130 :  
0000 131 :  
0000 132 :  
0000 133 :  
0000 134 :  
0000 135 :  
0000 136 :  
0000 137 :  
0000 138 :  
0000 139 :  
0000 140 :  
0000 141 :  
0000 142 :  
0000 143 :  
0000 144 :  
0000 145 :  
0000 146 :  
0000 147 :  
0000 148 :  
0000 149 :  
0000 150 :  
0000 151 :  
0000 152 :  
0000 153 :  
0000 154 :  
0000 155 :  
0000 156 :  
0000 157 :  
0000 158 :  
0000 159 :  
0000 160 :  
0000 161 :  
0000 162 :  
0000 163 :  
0000 164 :  
0000 165 :  
0000 166 :  
0000 167 :  
0000 168 :  
0000 169 :  
0000 170 :  
0000 171 :

is required, the routines which perform the formatting make use of the information stored in this UCB location.

Three types of information maybe required for additional formatting - device registers, a remote port number, or CI packet information. I have defined a OPA0 error logging control flag for each information type. For a given error condition the setting of these control flags will direct what information is saved within this new UCB location, before the fork process is created, to be used in the formatting of the appropriate \_OPA0 error log message.

V03-028 TMK0003 Todd M. Katz 06-Mar-1984  
Add support for \_OPA0 error logging. This involves determining, whenever error logging is to be done, whether or not an attempt should also be made to log the error condition at \_OPA0. Such logging will always be attempted for certain error conditions, and it will also be done whenever it is found that the system device, which is presumed to also be the error logging device, is currently unavailable.

A table driven routine, OPA0\_LOG, is used to determine whether or not \_OPA0 error logging should always be done for a given error condition as well as to provide the error logging message to be broadcast to \_OPA0 and optional formatting information. When a decision is made to perform this error logging, the UCB's message fork block is used to create a fork process provided it is not already in use (in which case \_OPA0 error logging will be bypassed for this error condition). When this fork process resumes control at OPA0\_LOG\_FORK, it proceeds to format an error logging message and broadcast it to \_OPA0. In the case of certain unrecoverable port initialization errors, this fork process will also broadcast a second message indicating that the port will be left offline.

V03-027 TMK0002 Todd M. Katz 21-Feb-1984  
Make the following changes to fix several bugs, and in support of allowing port initialization to proceed at IPL 8 instead of at IPL\$POWER:

1. Do not disable all interrupts by raising IPL to IPL\$POWER before calling INISPORT from within ERR\$INIPORT. Port initialization is now being done at fork IPL instead of at IPL\$POWER.
2. Disable device interrupts within ERR\$INIPORT before calling INISPORT to re-initialize the port. This is done by explicitly placing the port within the un-initialized state. If this is not done it is possible that the port maybe in the un-initialized state but with device interrupts enabled when port re-initialization begins. Then if a device interrupt occurs during port re-initialization it may prevent the un-initialized -> disabled state transition from occurring at the proper time. The end result is that a second attempt at re-initializing the port will be required.
3. The way in which ERR\$PWF\_RECOV is forking is incorrect.

0000 172 :  
0000 173 :  
0000 174 :  
0000 175 :  
0000 176 :  
0000 177 :  
0000 178 :  
0000 179 :  
0000 180 :  
0000 181 :  
0000 182 :  
0000 183 :  
0000 184 :  
0000 185 :  
0000 186 :  
0000 187 :  
0000 188 :  
0000 189 :  
0000 190 :  
0000 191 :  
0000 192 :  
0000 193 :  
0000 194 :  
0000 195 :  
0000 196 :  
0000 197 :  
0000 198 :  
0000 199 :  
0000 200 :  
0000 201 :  
0000 202 :  
0000 203 :  
0000 204 :  
0000 205 :  
0000 206 :  
0000 207 :  
0000 208 :  
0000 209 :  
0000 210 :  
0000 211 :  
0000 212 :  
0000 213 :  
0000 214 :  
0000 215 :  
0000 216 :  
0000 217 :  
0000 218 :  
0000 219 :  
0000 220 :  
0000 221 :  
0000 222 :  
0000 223 :  
0000 224 :  
0000 225 :  
0000 226 :  
0000 227 :  
0000 228 :

It does not make proper use of the UCB\_V\_FKLOCK fork block interlock bit. It never sets the interlock bit before using the fork block if the fork block is currently not in use. This may result in this same fork block being used twice in succession. In such a situation the context saved by the first fork, the fork initiated by ERR\$PWF\_RECOV, would be overwritten by the context of the second fork.

I have corrected this problem by utilizing the new routine INIS\$FORK to control the forking. This routine knows how to extract the fork block from the appropriate fork queue in an atomic fashion, and how to make proper use of the fork block interlock bit. This routine always returns control at fork IPL by jumping to the address provided it as input in R3.

4. I have also corrected an error in how ERR\$PWF\_RECOV cleans up a local port's path blocks, and crashes the local port. This routine should only be crashing the port after every SYSAP with a connection over the port has been notified and has had a chance to issue a DISCONNECT. A DISCONNECT, under such a circumstance, would result in the path block being deleted, and the count of path blocks associated with the port being decremented, if the disconnected connection represented the path's last connection. Therefore, ERR\$PWF\_RECOV should only be crashing the port when the count of path blocks associated with the port reaches zero indicating that every SYSAP which had a connection over this port has been notified and issued a DISCONNECT.

Unfortunately when the co-routine CNF\$LKP\_PB\_PDT encounters the end of the PB list, ERR\$PWF\_RECOV immediately crashes the port regardless of the number of path blocks still associated with the port. I have corrected this routine so that when the end of the port's path block list is encountered, ERR\$PWF\_RECOV will only crash the port if the count of the port's associated path blocks is zero.

V03-026 TMK0001

Todd M. Katz

14-Feb-1984

Add support for error logging of the refusals of the local port to open up a virtual circuit to a remote port because of confliotions between information provided by the remote system and a known system within the system-wide configuration data base. This support involves modification to ELOG\$PACKET so that a special type of packet is logged whenever this event occurs. Instead of logging a data packet, this event results in the logging of the known system ID, the known system nodename, and the remote system nodename in addition to the usual stuff which is always logged (local station address, etc... ).

Also, fix two small bugs within ELOG\$PACKET. Currently, the entire message logging area is not being used (or is not being zeroed out if there is no packet to be logged). This is because the destination sizes used in the MOVCSs only include 4 bytes of the 8 bytes of CI packet command/control/status information, CI packet PPD type, and CI packet message data length.



0000	229	:	V03-025	NPK3044	N. Kronenberg	6-Feb-1984
0000	230	:		Add ELOG\$ERROR_DG to log an error datagram. Modify		
0000	231	:		ELOG\$\$LOG_LM to handle error log datagrams which are		
0000	232	:		larger than other logged messages.		
0000	233	:		Disable all optional bugchecks in ERR\$DEBUGCHECK.		
0000	234	:				
0000	235	:	V03-024	NPK3043	N. Kronenberg	6-Feb-1984
0000	236	:		Fix ELOG\$\$LOG_LM to copy all 6 bytes of local sysid.		
0000	237	:				
0000	238	:	V03-023	NPK3039	N. Kronenberg	11-Jan-1984
0000	239	:		Zero PB\$L_CLSCKT_DG when closing vc in ERR\$CRASHVC.		
0000	240	:		Add ERR\$V_DEB_PSRX flag for enabling/disabling bugcheck		
0000	241	:		on interrupt with undefined bits set in PSR.		
0000	242	:				
0000	243	:	V03-022	NPK3038	N. Kronenberg	6-Dec-1983
0000	244	:		Disable the ERR\$DEBUGCHECK flags for connect request		
0000	245	:		with no path block and SCS bookkeeping with no path		
0000	246	:		block.		
0000	247	:				
0000	248	:	V03-021	NPK3037	N. Kronenberg	11-Nov-1983
0000	249	:		Add ERR\$DEBUGCHECK flags definitions and flags longwd.		
0000	250	:		Make subroutine CLEANUP_PKTS a global routine,		
0000	251	:		ERR\$CLEANUP_PKT.		
0000	252	:		Make subroutine CALL_INIT_PORT a global routine,		
0000	253	:		ERR\$INIPOINT.		
0000	254	:		Remove queue interlock clear from FLUSH_Q since it		
0000	255	:		is already done in routine UNLOCK_BADQ.		
0000	256	:				
0000	257	:	V03-020	NPK3029	N. Kronenberg	22-Jul-1983
0000	258	:		Enhancements for V4.0:		
0000	259	:		Change ERR\$CRASH_PORT to not fake a power off to		
0000	260	:		prevent reinit of port if ERTCNT is exhausted		
0000	261	:		(INISPORT now handles that.)		
0000	262	:		Change IOFORK to FORK in ERR\$PWF_RECOV.		
0000	263	:		Remove references to PB\$L_SB in favor of PB\$L_SBLINK.		
0000	264	:				
0000	265	:	V03-019	NPK3024	N. Kronenberg	18-May-1983
0000	266	:		Add logic for variable net header size to routine		
0000	267	:		ELOG\$LOG_LM.		
0000	268	:				
0000	269	:	V03-018	KTA3046	Kerbey T. Altmann	30-Mar-1983
0000	270	:		Redo for SCS/PPD split.		
0000	271	:				
0000	272	:	V03-017	NPK3011	N. Kronenberg	22-Nov-1982
0000	273	:		Fix ERR\$CRASH_PORT to call ERR\$PWF_RECOV at device IPL.		
0000	274	:				
0000	275	:	V03-016	ROW0133	Ralph O. Weber	14-OCT-1982
0000	276	:		Correct PPD\$W_LENGTH reference in ELOG\$\$LOG_LM to PPD\$W_SIZE.		
0000	277	:		This causes the allocated pool size value to be used, as		
0000	278	:		documented, when the maximum size of the message region to be		
0000	279	:		error logged is calculated.		
0000	280	:		This change will be distributed in Version 3.2.		
0000	281	:				
0000	282	:	V03-015	NPK3006	N. Kronenberg	9-Sep-1982
0000	283	:		Comment possible aux status input to ERR\$PWF_RECOV better.		
0000	284	:		Fix data structure error path by zeroing locked queue		
0000	285	:		headers in ERR\$PWF_RECOV prior to forking down from		



```
0000 286 : device IPL.
0000 287 :
0000 288 :
0000 289 : V03-014 ROW0119 Ralph O. Weber 9-AUG-1982
0000 290 : Modify ELOG$$LOG LM so that it does not copy anything beyond
0000 291 : the space allocated to a message packet as shown in the size
0000 292 : word field of the standard pool unit header.
0000 293 : This change will be in a new driver image shipped in V3.1.
0000 294 :
0000 295 : V03-013 ROW0115 Ralph O. Weber 30-JUN-1982
0000 296 : Modify ELOG$$LOG LM to always copy first 68 bytes of message
0000 297 : into UCB logged message buffer and to specially zero the
0000 298 : buffer when no message packet exists. Also replace
0000 299 : ELOG$$LOG LM system block search code with use of new
0000 300 : P$SL_SBLINK pointer to SB.
0000 301 : This change will be in a new driver image shipped in V3.1.
0000 302 :
0000 303 : V03-12 NPK3001 N. Kronenberg 28-Jun-1982
0000 304 : Clear UCB fork blk lock following power fail fork.
0000 305 :
0000 306 : V03-011 ROW0111 Ralph O. Weber 27-JUN-1982
0000 307 : Add ELOG$CABLES, a routine like ELOG$PACKET only with change
0000 308 : of cable state error type. This routine required for loopback
0000 309 : datagram logging. Add a clear for UCB$C_ICMD when there is
0000 310 : no message packet so that it will be zero just like everything
0000 311 : else.
0000 312 : This change will be in a new driver image shipped in V3.1.
0000 313 :
0000 314 : V03-010 ROW0110 Ralph O. Weber 24-JUN-1982
0000 315 : Fix ELOG$$LOG LM to adjust error count up by one while copying
0000 316 : it into the UCB log message buffer, since UCB$W_ERRCNT has not
0000 317 : yet been incremented.
0000 318 : This change will be in a new driver image shipped in V3.1.
0000 319 :
0000 320 : V03-009 ROW0108 Ralph O. Weber 24-JUN-1982
0000 321 : Fix ELOG$PACKET and ELOG$$LOG LM to handle case where no
0000 322 : packet exists. Also correct ELOG$PACKET so that error sub-
0000 323 : type information is retrieved after CNF$LKP_PB_MSG is called.
0000 324 : This change will be shipped with VAX/VMS Version 3.1.
0000 325 :
0000 326 : V03-008 NPK3001 N. Kronenberg 22-Jun-1982
0000 327 : Fix to keep UCB fork block locked on power fail
0000 328 : recovery fork.
0000 329 :
0000 330 : V03-007 ROW0098 Ralph O. Weber 7-JUN-1982
0000 331 : Add call to error appropriate error logging routine at
0000 332 : CONFIG ERR in ERR$VCCLOSED MSG.
0000 333 : This change will be in a new driver image shipped in V3.1.
0000 334 :
0000 335 : V03-006 ROW0092 Ralph O. Weber 3-JUN-1982
0000 336 : Add error logging routines which generate logged message error
0000 337 : log entries; ELOG$PACKET, ELOG$PTH_ST_CHG, and ELOG$CBL_X_CHG.
0000 338 : Also added necessary definition macro references.
0000 339 : This change will be in a new driver image shipped in V3.1.
0000 340 :
0000 341 : V03-005 ROW0089 Ralph O. Weber 20-MAY-1982
0000 342 : Add error logging routines which generate device attention
error log entries; ELOG$INIT_SWERR, ELOG$UCODE_NORD,
```

```
0000 343 : ELOG$HARDWARE, and ELOG$INTRLOCK. Also add register dump
0000 344 : routine, ELOG$REGDUMP. Add necessary definition macro
0000 345 : references too.
0000 346 : This change will be in a new driver image shipped in V3.1.
0000 347 :
0000 348 : V03-004 NPK2019 N. Kronenberg 6-Apr-1982
0000 349 : Changed DISP_ENTRY to global ERR$DISP_ENTRY.
0000 350 : Add routine ERR$CRASH_PORT.
0000 351 : Fix illegal CDT state in NOTIFY_SYSAP to be nonfatal
0000 352 : bugcheck with recovery rather than fatal bugcheck.
0000 353 : Fix PB lookup failure in ERR$VCCLOSED MSG to crash VC.
0000 354 : Change queue interlock failure in FLUSH_Q to be non
0000 355 : fatal bugcheck.
0000 356 : Fix CHK_NO_CDTS to get remote port from PB and use
0000 357 : $TURNMSG.
0000 358 : Fix CLEANUP_PKTS to reset logout area longwd immediately
0000 359 : after processing entry.
0000 360 :
0000 361 : V03-003 NPK2018 N. Kronenberg 29-Mar-1982
0000 362 : Modified ERR$CRASHVC PB to use dg buffer in PB for
0000 363 : SETCKT instead of allocating buffer.
0000 364 : Broke ERR$DISC_VCFAIL into main routine and new
0000 365 : subroutine, CHK_NO_CDTS.
0000 366 : Made disconnect on power failure synchronous --
0000 367 : it suspends till CDT is actually removed.
0000 368 : Modified CONNECT_ABO and DCONNECT_OK in NOTIFY_SYSAP
0000 369 : to call CHK_NO_CDTS.
0000 370 :
0000 371 : V03-002 NPK2018 N. Kronenberg 25-Mar-1982
0000 372 : Fix ERR$DISC_PWFAIL to purge out command queues again.
0000 373 :
0000 374 : V03-001 NPK2016 N. Kronenberg 18-Mar-1982
0000 375 : Fixed .TITLE
0000 376 :
0000 377 :
0000 378 :--
```



```
0000 380      .SBTTL  DEFINITIONS
0000 381
0000 382  ::
0000 383  :: Set PSECT to driver code:
0000 384  ::
0000 385
0000 386      .PSECT  $$$115_DRIVER, LONG
0000 387
0000 388  ::
0000 389  :: System definitions (LIB.MLB)
0000 390  ::
0000 391
0000 392      .nocross
0000 393      $CDTDEF      : Connection Descriptor offsets
0000 394      $CLUBDEF   : Cluster Block offsets
0000 395      $CRBDEF    : Channel Request Block offsets
0000 396      $DDBDEF    : Device Data Block format
0000 397      $DYNDEF    : Dynamic data structures types
0000 398      $EMBDEF     : Error log buffer offsets
0000 399      $EMBLTDEF  : Logged messages subtype values
0000 400      $IDBDEF   : Interrupt Data Block offsets
0000 401      $IPLDEF   : Define priority levels
0000 402      $MCHKDEF  : Protect from machine check codes
0000 403      $PBDEF    : Path Blk offsets
0000 404      $PDTDEF    : Port Descriptor offsets
0000 405      $SBDEF     : System Block offsets
0000 406      $SSDEF     : System service return codes
0000 407      $UCBDEF   : UCB definitions
0000 408      $VECDEF   : CRB vector segment offsets
0000 409
0000 410  ::
0000 411  :: PADRIVER definitions (PALIB.MLB):
0000 412  ::
0000 413
0000 414      $PAERDEF     : PADRIVER error code definitions
0000 415      $PAPBDEF    : PA-specific extension to PB
0000 416      $PAPDTDEF  : PA-specific extension to PDT
0000 417      $PAREGDEF   : CI port device register defs
0000 418      $PAUCBDEF   : PA extension to UCB
0000 419      $PPDDEF      : PPD layer of msg/dg header
0000 420      .cross
```

```
0000 422      .SBTTL  _OPAO ERROR LOGGING DATA
0000 423
0000 424
0000 425  :+ The routine which logs errors to _OPAO is table driven. There are separate
0000 426  : tables for device attention and logged message errors. What follows is the
0000 427  : the macro that is used to generate each table entry, the two tables, various
0000 428  : offsets to the fields within each table entry, and assorted constants.
0000 429  :-
0000 430
0000 431
0000 432  : Macro to generate an entry within an _OPAO error logging table. The format
0000 433  : of each entry is as follows:
0000 434
0000 435      .BYTE  <ERROR SUBTYPE>
0000 436      .BYTE  <ERROR TYPE>
0000 437      .BYTE  <CONTROL FLAGS>
0000 438      .BYTE  <OPTIONAL OFFSET TO MSG FIELD TO BE FORMATTED>
0000 439      .WORD  <OPTIONAL OFFSET (from PASCTINIT) TO FORMATTING ROUTINE>
0000 440      .WORD  <OFFSET (from PASCTINIT) TO ERROR MSG>
0000 441
0000 442  : All of the _OPAO error messages are placed within their own PSECT. Each
0000 443  : _OPAO error logging table must be terminated by a word of -1.
0000 444
0000 445
0000 446      .MACRO  $OPAO_LOG      TYPE,SUBTYPE,FLAGS,FORMAT,MSG
0000 447
0000 448      .IF      NB      TYPE
0000 449      .BYTE  <PAERSK_ES_'SUBTYPE>      ; Error Subtype
0000 450      .BYTE  <PAERSK_ET_'TYPE>      ; Error Type
0000 451
0000 452      .IF      NB      FLAGS
0000 453      .BYTE  FLAGS      ; Flags affecting logging to OPAO
0000 454      .ENDC
0000 455      .IF      B      FLAGS
0000 456      .BYTE  0
0000 457      .ENDC
0000 458
0000 459      .IF      NB      FORMAT
0000 460      .BYTE  %LOCATE(<xx>,MSG)+11      ; Offset to field to be formatted
0000 461      .WORD  <FORMAT-PASCTINIT>      ; Optional formatting routine offset
0000 462      .ENDC
0000 463      .IF      B      FORMAT
0000 464      .BYTE  0
0000 465      .WORD  0
0000 466      .ENDC
0000 467
0000 468      .SAVE
0000 469      .PSECT  $$$110_MSGS
0000 470  $$$MSG_PTR =
0000 471      .ASCII  <CR><LF><BELL>'"%PAX0, 'MSG'"<CR><LF> ; Message to display at OPAO
0000 472      .RESTORE
0000 473      .WORD  <$$$MSG_PTR-PASCTINIT> ; OPAO msg offset
0000 474      .ENDC
0000 475
0000 476      .IF      B      TYPE
0000 477      .WORD  -1      ; -1 marks the end of the table
0000 478      .ENDC
```



PAERROR  
V04-001

Error Handling & Logging Routines H 13  
\_OPAO ERROR LOGGING DATA

16-SEP-1984 01:16:25 VAX/VMS Macro V04-00  
10-SEP-1984 01:16:10 [DRIVER.SRC]PAERROR.MAR;2

Page 10  
(3)

0000 479 .ENDM

```
0000 481
0000 482 :
0000 483 : Offsets to the various fields within a _OPA0 error logging table entry.
0000 484 :
0000 485 :
00000000 0000 486 SUBTYPE = 0 : Offset to Error subtype
00000001 0000 487 TYPE = 1 : Offset to Error type
00000002 0000 488 CFLAGS = 2 : Offset to Control Flags
00000003 0000 489 OFFSET = 3 : Offset to Optional Formatting Offset
00000004 0000 490 FORMAT = 4 : Offset to Optional Format Routine Offset
00000006 0000 491 MSG = 6 : Offset to Error Message Offset
0000 492 :
00000008 0000 493 OPA0_LOG_SIZE = 8 : _OPA0 Error Logging Table Entry Size
0000 494 :
0000 495 :
0000 496 : Define the bits within the control flags _OPA0 error logging table field.
0000 497 :
0000 498 :
00000000 0000 499 V_ALWAYS = 0 : Always print out this error message
00000001 0000 500 M_ALWAYS = 1 :
0000 501 :
00000001 0000 502 V_OFFLINE = 1 : Always print out a second message
00000002 0000 503 M_OFFLINE = 2 : (Port has gone Offline)
0000 504 :
00000002 0000 505 V_RPORT = 2 : Store the remote port number in the
00000004 0000 506 M_RPORT = 4 : _OPA0 error logging UCB data area
0000 507 :
00000003 0000 508 V_PKT = 3 : Store the CICMD packet information in
00000008 0000 509 M_PKT = 8 : the _OPA0 error logging UCB data area
0000 510 :
00000004 0000 511 V_REGS = 4 : Store the device registers in the
00000010 0000 512 M_REGS = 16 : _OPA0 error logging UCB data area
0000 513 :
0000 514 :
0000 515 : Define ASCII symbols for various hexadecimal formatting characters.
0000 516 :
0000 517 :
00000000 0000 518 CR = 13 : ASCII for carriage return,
0000000A 0000 519 LF = 10 : linefeed,
00000007 0000 520 BELL = 7 : and bell
0000 521 :
00000006 0000 522 CTRLR_NAME = 6 : Byte offset to device controller
0000 523 : letter in error logging messages
0000 524 :
0000 525 :
0000 526 : Define table for hexadecimal -> ASCII and hexadecimal -> decimal -> ASCII
0000 527 : conversions.
0000 528 :
0000 529 :
0000 530 CONV_TABLE:
42 41 39 38 37 36 35 34 33 32 31 30 0000 531 .ASCII /0123456789ABCDEF/
46 45 44 43 000C
```



```
0010 533
0010 534
0010 535 :: Device Attention _OPAO Error Logging Table.
0010 536 ::
0010 537
0010 538 DA_OPAO_LOG TAB:
0010 539 $OPAO_LOG INSW,POOL,M_ALWAYS+M_OFFLINE,,-
0010 540 <Insufficient Non-paged Pool for Initialization>
0018 541 $OPAO_LOG INSW,CODE,M_ALWAYS+M_OFFLINE,,-
0018 542 <Failed to Locate Port Micro-code Image>
0020 543 $OPAO_LOG INSW,SCSID,M_ALWAYS+M_OFFLINE,,-
0020 544 <SCSSYSTEMID has NOT been set to a Non-zero Value>
0028 545 $OPAO_LOG HW,UCDW,M_ALWAYS,,-
0028 546 <Micro-code Verification Error>
0030 547 $OPAO_LOG HW,INIT,M_ALWAYS+M_REGS,FORMAT_REGS,-
0030 548 <Port Transition Failure - CNF/PMC/PSR xxxxxxxx/xxxxxxx/xxxxxxx>
0038 549 $OPAO_LOG HW,HWER,M_ALWAYS+M_REGS,FORMAT_REGS,-
0038 550 <Port Error Bit(s) Set - CNF/PMC/PSR xxxxxxxx/xxxxxxx/xxxxxxx>
0040 551 $OPAO_LOG HW,PDWN,M_ALWAYS,,-
0040 552 <Port Power Down>
0048 553 $OPAO_LOG HW,PUP,M_ALWAYS,,-
0048 554 <Port Power Up>
0050 555 $OPAO_LOG HW,UXIN,M_ALWAYS+M_REGS,FORMAT_REGS,-
0050 556 <Unexpected Interrupt - CNF/PMC/PSR xxxxxxxx/xxxxxxx/xxxxxxx>
0058 557 $OPAO_LOG HW,REVER,M_ALWAYS,FORMAT_REV,-
0058 558 <CI port ucode not at required rev level. RAM/PROM rev is xxxx/xxxx>
0060 559 $OPAO_LOG HW,REVCA,M_ALWAYS,FORMAT_REV,-
0060 560 <CI port ucode not at current rev level. RAM/PROM rev is xxxx/xxxx>
0068 561 $OPAO_LOG HW,CPUREV,M_ALWAYS,,-
0068 562 <CPU ucode not at required rev level for CI activity>
0070 563 $OPAO_LOG ILCK,MQRM,M_ALWAYS,,-
0070 564 <Message Free Queue Remove Failure>
0078 565 $OPAO_LOG ILCK,DQRM,M_ALWAYS,,-
0078 566 <Datagram Free Queue Remove Failure>
0080 567 $OPAO_LOG ILCK,RQRM,M_ALWAYS,,-
0080 568 <Response Queue Remove Failure>
0088 569 $OPAO_LOG ILCK,HCIN,M_ALWAYS,,-
0088 570 <High Priority Command Queue Insert Failure>
0090 571 $OPAO_LOG ILCK,LCIN,M_ALWAYS,,-
0090 572 <Low Priority Command Queue Insert Failure>
0098 573 $OPAO_LOG ILCK,MQIN,M_ALWAYS,,-
0098 574 <Message Free Queue Insert Failure>
00A0 575 $OPAO_LOG ILCK,DQIN,M_ALWAYS,,-
00A0 576 <Datagram Free Queue Insert Failure>
00A8 577 $OPAO_LOG
```

```
00AA 579
00AA 580
00AA 581 : : : : :
00AA 582 : : : : :
00AA 583 : : : : :
00AA 584 LM_OPAO_LOG_TAB:
00AA 585 $OPAO_LOG PKT,UPKT,M_ALWAYS+M_PKT,FORMAT PKT,-
00AA 586 <Unrecognized SCA Packet - FLAGS/OPC/STATUS/PORT xx/xx/xx/xx>
00B2 587 $OPAO_LOG PKT,PCVC,M_ALWAYS+M_RPORT,FORMAT PORT,-
00B2 588 <Port has Closed Virtual Circuit - REMOTE PORT xxx>
00BA 589 $OPAO_LOG PKT,CSHP,M_ALWAYS,-
00BA 590 <Software Shutting Down Port>
00C2 591 $OPAO_LOG PKT,SCVC,M_ALWAYS+M_RPORT,FORMAT PORT,-
00C2 592 <Software is Closing Virtual Circuit - REMOTE PORT xxx>
00CA 593 $OPAO_LOG PKT,CNPB,M_ALWAYS+M_PKT,FORMAT PKT,-
00CA 594 <Received Connect Without Path-Block - FLAGS/OPC/STATUS/PORT xx/xx/
00D2 595 $OPAO_LOG PKT,SCA,M_ALWAYS+M_PKT,FORMAT PKT,-
00D2 596 <Inappropriate SCA Control Message - FLAGS/OPC/STATUS/PORT xx/xx/xx
00DA 597 $OPAO_LOG PKT,NOPB,M_ALWAYS+M_RPORT,FORMAT PORT,-
00DA 598 <No Path-Block During Virtual Circuit Close - REMOTE PORT xxx>
00E2 599 $OPAO_LOG PKT,ERRDG,M_RPORT,FORMAT PORT,-
00E2 600 <HSC Error Logging Datagram Received - REMOTE PORT xxx>
00EA 601 $OPAO_LOG PKT,RSCKS,M_ALWAYS+M_RPORT,FORMAT PORT,-
00EA 602 <Remote System Conflicts with Known System - REMOTE PORT xxx>
00F2 603 $OPAO_LOG CBL,OGB,M_RPORT,FORMAT PORT,-
00F2 604 <Path #0. Has gone from GOOD to BAD - REMOTE PORT xxx>
00FA 605 $OPAO_LOG CBL,IGB,M_RPORT,FORMAT PORT,-
00FA 606 <Path #1. Has gone from GOOD to BAD - REMOTE PORT xxx>
0102 607 $OPAO_LOG CBL,OBG,M_RPORT,FORMAT PORT,-
0102 608 <Path #0. Has gone from BAD to GOOD - REMOTE PORT xxx>
010A 609 $OPAO_LOG CBL,IBG,M_RPORT,FORMAT PORT,-
010A 610 <Path #1. Has gone from BAD to GOOD - REMOTE PORT xxx>
0112 611 $OPAO_LOG CBL,UC,M_RPORT,FORMAT PORT,-
0112 612 <Cables have gone from UNCROSSED to CROSSED - REMOTE PORT xxx>
011A 613 $OPAO_LOG CBL,CU,M_RPORT,FORMAT PORT,-
011A 614 <Cables have gone from CROSSED to UNCROSSED - REMOTE PORT xxx>
0122 615 $OPAO_LOG CBL,LOGB,M_ALWAYS,-
0122 616 <Path #0. Loopback has gone from GOOD to BAD>
012A 617 $OPAO_LOG CBL,LIGB,M_ALWAYS,-
012A 618 <Path #1. Loopback has gone from GOOD to BAD>
0132 619 $OPAO_LOG CBL,LOBG,M_ALWAYS,-
0132 620 <Path #0. Loopback has gone from BAD to GOOD>
013A 621 $OPAO_LOG CBL,LIBG,M_ALWAYS,-
013A 622 <Path #1. Loopback has gone from BAD to GOOD>
0142 623 $OPAO_LOG CBL,LOBX,M_RPORT,FORMAT PORT,-
0142 624 <Path #0. Has become working but CROSSED to Path #1. - REMOTE PORT x
014A 625 $OPAO_LOG CBL,LIBX,M_RPORT,FORMAT PORT,-
014A 626 <Path #1. Has become working but CROSSED to Path #0. - REMOTE PORT x
0152 627 $OPAO_LOG
```



```
0154 629 .SBTTL ERR$CRASHVC, CRASH VC ON SPECIFIED
0154 630 .SBTTL - PATH BLOCK
0154 631
0154 632
0154 633 :+
0154 634 : These routines are called to crash an open virtual circuit on
0154 635 : a specific path. ERR$CRASHVC sets VC failure in progress
0154 636 : status in the PB and does a SETCKT closed to the remote port. Return
0154 637 : is then taken since the SETCKT response will continue the process of
0154 638 : cleaning up the broken VC.
0154 639 :
0154 640 : In case the response pkt is a REQID or other datagram type pkt,
0154 641 : there may be no path block. In this case, return is taken without
0154 642 : doing anything.
0154 643 :
0154 644 : Inputs:
0154 645 :
0154 646 :     IPL -Fork IPL
0154 647 :     R1 -Addr of PB
0154 648 :     R2 -Addr of msg/dg response
0154 649 :     R4 -PDT addr
0154 650 :
0154 651 :     VC state -open
0154 652 :
0154 653 : Outputs:
0154 654 :
0154 655 :     R0-R1 -Destroyed
0154 656 :     Other registers -Preserved; in particular, the msg/dg
0154 657 :                     pointed to by R2 is not disposed of --
0154 658 :                     that is the caller's responsibility
0154 659 :
0154 660 :
0154 661 : .ENABL LSB
0154 662 :
0154 663 ERR$CRASHVC::
0154 664 :
0154 665 : TSTL R1 ; Got a valid path block?
0154 666 : BEQL 20$ ; No, just leave
0154 667 : PUSHL R2 ; Save caller's R2
0154 668 : CMPW PB$W_STATE(R1),- ; Is virtual circuit failure
0154 669 : #PB$C_VC_FAIL ; already in progress?
0154 670 : BEQL 10$ ; Branch if so
0154 671 : MOVW #PB$C_VC_FAIL,- ; Set VC failure in progress
0154 672 : PB$W_STATE(R1) ; on this PB
0154 673 : MOVL PB$L_CLSCKT_DG(R1),R2 ; Get addr of SETCKT dg in PB
0154 674 : CLRL PB$L_CLSCKT_DG(R1) ; Zero dg address to show that port
0154 675 : ; owns pkt now
0154 676 : BISL3 #<PPD$M_RSPa24>,- ; Tell port to mark VC closed
0154 677 : <PPD$C_SETCKTa16>,- ;
0154 678 : PB$B_RSTATION(R1),- ; to this remote station
0154 679 : PPD$B_PURT(R2) ; Do SETCKT at top priority
0154 680 : MOVZWL #PPD$M_CST,PPD$W_MASK(R2) ; to close VC
0154 681 : CLRL PPD$W_M_VAL(R2) ; Get response to reclaim buffer
0154 682 : MOVW #PPD$M_DISPOSE,- ;
0154 683 : PPD$B_SWFLAG(R2) ; Ask interrupt serv to notify us
0154 684 : BSBW INT$INS_COMQH ; Do it
0154 685 : 10$: POPL R2 ; Restore caller's R2
```

PAERROR  
V04-001

Error Handling & Logging Routines M 13  
- PATH BLOCK

16-SEP-1984 01:16:25 VAX/VMS Macro V04-00  
10-SEP-1984 01:16:10 [DRIVER.SRC]PAERROR.MAR;2

Page 15  
(10)

05 018C 686 20\$: RSB ; Return  
018D 687  
018D 688 .DSABL LSB

```
018D 690 .SBTTL ERR$CRASH_PORT, INIT PORT CRASH
018D 691
018D 692
018D 693 :+
018D 694 ERR$CRASH_PORT is called by the driver at fork IPL detecting an error
018D 695 which might be either a software error or a port hardware or firmware error.
018D 696
018D 697 Action is to maintenance init the port to prevent further activity,
018D 698 and, if there are any error retries left, to call ERR$PWF_RECOV
018D 699 in simulation of a power fail recovery. If no retries are left,
018D 700 then PUP is cleared in PDT$W_LPORT_STS to prevent the port from being
018D 701 reinitialized. ERR$PWF_RECOV initiates a fork process on the UCB
018D 702 which takes care of notifying SYSAPs and cleaning up the configuration
018D 703 database eventually. The main difference between deliberately crashing
018D 704 the port and a real power failure is that in the crash case, cached
018D 705 packets are not written to the logout area by the port and hence may not
018D 706 be reclaimed.
018D 707
018D 708 Inputs:
018D 709 R4 -PDT addr
018D 710 (SP) -Caller's PC
018D 711
018D 712 Outputs:
018D 713 R0,R1 -Destroyed
018D 714
018D 715 Other registers -Preserved
018D 716
018D 717 :-
018D 718
018D 719
018D 720 .ENABL LSB
018D 721
018D 722 ERR$CRASH_PORT::
018D 723
018D 724 BBSS #PDT$V_PWF_CLNUP,- : Set PWF cleanup in progress
018D 725 PDT$W_LPORT_STS(R4),20$ : Branch if set already
018F 726 PUSHB #M<R2,R3,R4,R5> : Save registers
0193 727 MOVL #PA_PMC_M_MIN,- : Maintenance init the port
0195 728 @PDT$W_LPORT_STS(R4)
0197 729 MOVL PDT$W_UCB0(R4),R5 : Get UCB addr
019A 730 BICW #UCB$M_ONLINE,- : Set unit offline to show init
019F 731 UCB$W_STS(R5) : in progress
01A1 732 MOVZWL #SS$_ABORT,R1 : Assume we have more retries,
01A3 733 : but let SYSAP know not to
01A6 734 : expect cached entries back
01A6 735 DECB UCB$B_ERTCNT(R5) : Decr retry count
01AA 736 BGEQ 10$ : Branch if not out of retries
01AC 737 MOVZWL #SS$_CTRLERR,R1 : Else set aux status to tell
01B1 738 : SYSAP's port won't return
01B1 739
01B1 740 10$: DSBINT UCB$B_DIPL(R5) : Set IPL up to device to block
01B8 741 : interrupts
01B8 742 BSBW ERR$PWF_RECOV : Treat like power failure from here on
01BB 743 ENBINT : Restore IPL to fork IPL
01BE 744 POPR #M<R2,R3,R4,R5> : Restore registers
01C0 745
01C0 746 20$: RSB : Return to caller
```

2D 0110 00 E2 018D 724 BBSS #PDT\$V\_PWF\_CLNUP,- : Set PWF cleanup in progress  
C4 018F 725 PDT\$W\_LPORT\_STS(R4),20\$ : Branch if set already  
3C BB 0193 726 PUSHB #M<R2,R3,R4,R5> : Save registers  
01 D0 0195 727 MOVL #PA\_PMC\_M\_MIN,- : Maintenance init the port  
00E8 D4 0197 728 @PDT\$W\_LPORT\_STS(R4)  
55 00DC C4 D0 019A 729 MOVL PDT\$W\_UCB0(R4),R5 : Get UCB addr  
10 AA 019F 730 BICW #UCB\$M\_ONLINE,- : Set unit offline to show init  
64 A5 01A1 731 UCB\$W\_STS(R5) : in progress  
51 2C 3C 01A3 732 MOVZWL #SS\$\_ABORT,R1 : Assume we have more retries,  
: but let SYSAP know not to  
: expect cached entries back  
0080 C5 97 01A6 735 DECB UCB\$B\_ERTCNT(R5) : Decr retry count  
05 18 01AA 736 BGEQ 10\$ : Branch if not out of retries  
51 0054 8F 3C 01AC 737 MOVZWL #SS\$\_CTRLERR,R1 : Else set aux status to tell  
: SYSAP's port won't return  
01B1 738  
01B1 739  
01B1 740 10\$: DSBINT UCB\$B\_DIPL(R5) : Set IPL up to device to block  
: interrupts  
0006 30 01B8 742 BSBW ERR\$PWF\_RECOV : Treat like power failure from here on  
: Restore IPL to fork IPL  
3C BA 01BB 743 ENBINT : Restore IPL to fork IPL  
: Restore registers  
05 01C0 745  
01C0 746 20\$: RSB : Return to caller



PAERROR  
V04-001

Error Handling & Logging Routines B 14  
ERR\$CRASH\_PORT, INIT PORT CRASH

16-SEP-1984 01:16:25 VAX/VMS Macro V04-00  
10-SEP-1984 01:16:10 [DRIVER.SRC]PAERROR.MAR;2

Page 17  
(11)

01C1 747  
01C1 748 .DSABL LSB

```
01C1 750 .SBTTL ERR$PWF_RECOV, NOTIFY SYSAPS WITH
01C1 751 .SBTTL - CONNECTIONS ON POWER
01C1 752 .SBTTL - FAILED PORT
01C1 753
01C1 754
01C1 755 :+ ERR$PWF_RECOV is called by unit initialization on power fail
01C1 756 : recovery or by port interrupt service on power down or by ERR$CRASH_PORT.
01C1 757 : ERR$PWF_RECOV first checks for packet queues that might be corrupted
01C1 758 : and for corrupted queues zeros the queue header, thus preventing
01C1 759 : future attempts to remove entries for the queue and causing bugchecks.
01C1 760 : ERR$PWF_RECOV then forks to lower IPL to the SCS synchronization
01C1 761 : level. Next, all formative path blocks on this PDT (i.e., START
01C1 762 : handshakes in progress) are looked up and formative PB's and SB's are
01C1 763 : deallocated to pool.
01C1 764
01C1 765 : ERR$PWF_RECOV then calls CNF$LKP_PB PDT to look up PB's associated with
01C1 766 : the failed PDT. CNF$LKP_PB PDT calls us back as a coroutine for each
01C1 767 : PB found. For each PB, the CDT list is searched and, for each open CDT,
01C1 768 : the SYSAP error address is called with appropriate status. SYSAP
01C1 769 : DISCONNECTs issued as a result of error routines being called continue
01C1 770 : the failure process. (See routine ERR$DISC_PWFAIL for more info.)
01C1 771
01C1 772 : CDT's in non-open states are handled the same as described in ERR$VCCLOSED_MSG.
01C1 773
01C1 774 : There is a difference between connection cleanup following a VC failure
01C1 775 : and connection cleanup following a port failure. In the VC failure case,
01C1 776 : the port is still alive. As sysap's with connections on the broken vc
01C1 777 : are notified and issue disconnects, CDT's are retained in the PB CDT list.
01C1 778 : They are retained because queued traffic may still be in the port which will
01C1 779 : be completing with appropriate status. The CDT's are cleaned up after the
01C1 780 : last one is disconnected and after the cache clear msg has made it through
01C1 781 : the port.
01C1 782
01C1 783 : If the vc is breaking because of a port failure, the port is dead and
01C1 784 : no further traffic will be processed. In this case, as sysap's disconnect,
01C1 785 : CDT's are cleaned up immediately. (Implementation note: this logic
01C1 786 : might be simplified overall by handling the simpler port crash case
01C1 787 : like the more complex vc failure case. The two cases probably need to
01C1 788 : differ only in their dependency on the cache clear msg.)
01C1 789
01C1 790 : Given the difference in handling, a problem occurs if a port crash
01C1 791 : happens in the midst of a vc failure. The port crash always results in
01C1 792 : a call to ERR$PWF_RECOV which forks prior to processing all the path
01C1 793 : blocks. Consequently, the code which notifies all sysap's in the
01C1 794 : event of a vc failure is not interrupted by the code in ERR$PWF_RECOV
01C1 795 : which processes PB's. When we arrive at the point of processing each
01C1 796 : PB, we are in one of two situations if the PB is in VC_FAIL state:
01C1 797
01C1 798 : -All CDT's are in VC_FAIL state also, and a cache clear has
01C1 799 : been issued which we have just cleaned up of one of the queues.
01C1 800
01C1 801 : -Some CDT's are in VC_FAIL. Sysap's have all been notified
01C1 802 : about the rest of the connections, but have not yet disconnected.
01C1 803
01C1 804 : So, if the PB is already in VC_FAIL state, CDT's in VC_FAIL state are
01C1 805 : closed out after completing the pending disconnect calls. If no CDT's
01C1 806 : remain after this, PB (and SB) are also deleted and port reinit may
```

```
01C1 807 : be attempted. If some CDT's remain, then place the PB in the PWR_FAIL
01C1 808 : state so that the remaining disconnects behave properly (like a port
01C1 809 : failure rather than a vc failure.)
01C1 810 :
01C1 811 : Inputs:
01C1 812 :
01C1 813 :     IPL                                -IPL$_POWER, device IPL
01C1 814 :
01C1 815 :     R1                                -Aux status to report to SYSAP:
01C1 816 :
01C1 817 :     SSS POWERFAIL if called by unit init
01C1 818 :     following CPU pwr fail recovery;
01C1 819 :
01C1 820 :     SSS POWERFAIL if called by int service
01C1 821 :     on port pwr down;
01C1 822 :
01C1 823 :     SSS ABORT if called by int service or
01C1 824 :     ERR$CRASH_PORT with error necessitating
01C1 825 :     reinit of port (buffers cached by port lost);
01C1 826 :
01C1 827 :     SSS CTRLERR if called by int service or
01C1 828 :     ERR$CRASH_PORT with error necessitating
01C1 829 :     reinit of port, but no retries are left
01C1 830 :     so that the port will remain shutdown
01C1 831 :     (buffers cached by port lost).
01C1 832 :
01C1 833 :     R5                                -UCB 0 addr
01C1 834 :
01C1 835 :     Port state                        -Uninitialized/maint; PDT/PQB
01C1 836 :                                         logout area contains a list of
01C1 837 :                                         port cached entries.
01C1 838 :
01C1 839 :     PDT$W_LPORT_STS                  -PWF_CLNUP set to show powerfail
01C1 840 :                                         cleanup in progress.
01C1 841 :                                         PUP set if called from system
01C1 842 :                                         powerfail recovery to show power up.
01C1 843 :                                         PUP clear if called from port interrupt
01C1 844 :                                         on power down to show power not
01C1 845 :                                         recovered yet.
01C1 846 :
01C1 847 :     (SP)                             -Return to caller in unit initialization
01C1 848 :                                         or interrupt service.
01C1 849 :
01C1 850 : Outputs:
01C1 851 :
01C1 852 :     IPL                                -IPL --> IPL$ SCS and return taken to
01C1 853 :                                         unit init; The unit is set offline
01C1 854 :                                         and registers preserved on return to
01C1 855 :                                         unit init.
01C1 856 :
01C1 857 :
01C1 858 :     .ENABL LSB
01C1 859 :
01C1 860 : ERR$PWF_RECOV::
01C1 861 :
01C1 862 :     BICW #UCB$M_ONLINE,-                ; Set unit offline to show
01C3 863 :     UCB$W_STS(R5)                       ; that it's uninitialized
```

10 AA  
64 A5



```
54 0084 C5 D0 01C5 864
53 01E0 C4 DE 01CA 865      MOVL   UCBSL_PDT(R5),R4      : Get PDT addr
      52 D4 01CF 866      MOVAL  PDT$Q_COMQBASÉ(R4),R3 : Get addr of 1st command queue hdr
      01D1 867      CLRL   R2              : Zero count of command + rsp queues
      01D1 868
      01D1 869 10$:      BSBW   UNLOCK_BADQ      : Unlock and handle bad queue
53 00C1 30 01D1 870      ADDL   #8,R3          : Step to next queue hdr
      08 C0 01D4 871      AOBLEQ #<<PDT$Q_RSPQ - PDT$Q_COMQBASÉ>/8>,-
      04 F3 01D7 872      R2,10$          : Branch if more queues to check
      F6 52 01D9 873      MOVL   PDT$L_MFQHDR(R4),R3 : Get addr of free msg queue hdr
53 020C C4 D0 01DB 874      BSBW   UNLOCK_BADQ      : Check it
      00B2 30 01E0 875      MOVL   PDT$L_DFQHDR(R4),R3 : Get addr of free dg queue hdr
53 0208 C4 D0 01E3 876      BSBW   UNLOCK_BADQ      : Check it
      00AA 30 01E8 877
      01EB 878
      54 51 D0 01EB 879      MOVL   R1,R4          : Copy aux status to reg preserved
      01EE 880      : that will be reserved over fork
53 000001F8'EF 9E 01EE 881      MOVAB  15$,R3      : Address of where to resume after fork
      FE08' 31 01F5 882      BRW    INIS$FORK      : Fork...
      01F8 883
      01F8 884
      01F8 885 : Clean up formative path and system blocks on this PDT. From this point on
      01F8 886 : we are at fork IPL.
      01F8 887
      01F8 888
54 0084 C5 D0 01F8 889 15$:      MOVL   UCBSL_PDT(R5),R4      : Restore PDT address
52 0174 C4 7E 01FD 890      MOVAQ  PDT$Q_FORMPB(R4),R2 : Get addr of formative PB
      0202 891      : listhead
53 62 D0 0202 892      MOVL   (R2),R3          : Get next formative PB
      0205 893
52 53 D1 0205 894 20$:      CMPL   R3,R2          : Back at listhead?
      21 13 0208 895      BEQL   50$          : Branch if so
50 30 A3 D0 020A 896      MOVL   PB$S_SBLINK(R3),R0 : Else get formative SB
      06 13 020E 897      BEQL   30$          : Branch if no SB
00000000'GF 16 0210 898      JSB    G^COM$DRVDEALMEM : Else deallocate SB to pool
      0216 899
      0C A3 E5 0216 900 30$:      BBCC   PB$B_RSTATION(R3),- : Turn off known port bit in
00 0114 C4 0219 901      PDT$B_PORTMAP(R4),40$ : bitmap
      50 53 D0 021D 902 40$:      MOVL   R3,R0          : Copy PB addr for deallocator
      53 63 D0 0220 903      MOVL   (R3),R3      : Get address of next formative PB
00000000'GF 16 0223 904      JSB    G^COM$DRVDEALMEM : Deallocate PB to pool
      DA 11 0229 905      BRB    20$          : Go for next formative PB
      022B 906
      62 52 D0 022B 907 50$:      MOVL   R2,(R2)      : Set formative pathblock
04 A2 52 D0 022E 908      MOVL   R2,4(R2)      : to empty
      0232 909
      0232 910
      0232 911 : Remove all packets from port command queues, response queue,
      0232 912 : free queues, and the logout area. All packets are returned to,
      0232 913 : pool except send datagrams which are flagged 'return to sysap.'
      0232 914 : These are returned to the SYSAP just as if they had gone out
      0232 915 : normally.
      0232 916
      0232 917
      00A7 30 0232 918      BSBW   ERR$CLEANUP_PKT      : Call packet cleanup routine
      0235 919
      0235 920 :
```

```
0235 921 : Clean up fully open paths and system blocks on this PDT:
0235 922 :
0235 923 :
FDCB' 30 0235 924 BSBW CNF$LKP_PB_PDT : Look up 1st/next PB
0235 925 : Start of coroutine if PB found:
48 50 E9 0238 926 BLBC R0,115$ : Branch if no more PB's
8000 BF B1 0238 927 CMPW #PB$C_VC_FAIL,- : Is PB already cleaning up a
12 A3 023F 928 PB$W_STATE(R3) : vc failure?
32 12 0241 929 BNEQ 100$ : Branch if not
0243 930 :
50 C8 A3 DE 0243 931 60$: MOVAL PB$L_CDTLST-CDT$L_CDTLST(R3),R0 : Else set to scan all CDT's on PB
0247 932 :
0247 933 :
50 6C A0 D0 0247 934 70$: MOVL CDT$L_CDTLST(R0),R0 : Get next CDT
0248 935 :
1C 13 0248 936 80$: BEQL 90$ : Branch if no more
28 A0 B1 024D 937 CMPW CDT$W_STATE(R0),- : SYSAP finished with connection?
OC 0250 938 #CDT$C_VC_FAIL : (I.e., disconnect issued?)
F4 12 0251 939 BNEQ 70$ : Branch if not
6C A0 DD 0253 940 PUSHL CDT$L_CDTLST(R0) : Save pointer to next CDT
53 DD 0256 941 PUSHL R3 : Save PB addr
53 50 D0 0258 942 MOVL R0,R3 : Put current CDT addr in standard reg
50 01 3C 025B 943 MOVZWL #SS$ NORMAL,R0 : Set status = success
FD9F' 30 025E 944 BSBW SCSS$CLOSE_CDT : Complete SYSAP's pending disconnect
0261 945 : call and deallocate CDT
53 8ED0 0261 946 POPL R3 : Retrieve PB address
50 8ED0 0264 947 POPL R0 : and addr of following CDT
E2 11 0267 948 BRB 80$ : Process next CDT, if any
0269 949 :
4000 8F B0 0269 950 90$: MOVW #PB$C_PWR_FAIL,- : Change PB state to power fail
12 A3 026D 951 PB$W_STATE(R3) : recovery/port failure in progress
34 A3 D5 026F 952 TSTL PB$L_CDTLST(R3) : All CDT's gone?
OC 13 0272 953 BEQL 110$ : Branch if so
05 0274 954 RSB : Else done -- remaining CDT's
0275 955 : will be cleaned up via disconnect calls
0275 956 :
4000 8F B0 0275 957 100$: MOVW #PB$C_PWR_FAIL,- : Set PB state to pwr fail
12 A3 0279 958 PB$W_STATE(R3) : in progress
34 A3 D5 027B 959 TSTL PB$L_CDTLST(R3) : Does this PB have any connections?
OD 12 027E 960 BNEQ 120$ : Branch if so
0280 961 :
FD7D' 30 0280 962 110$: BSBW CNF$REMOVE_PB : Else kill of this PB
0283 963 :
0112 C4 B5 0283 964 115$: TSTW PDT$W_PBCOUNT(R4) : Any PB's left on this PDT?
OB 12 0287 965 BNEQ 130$ : Branch if so, can't clean up port
00C9 30 0289 966 BSBW ERR$INIPOINT : Try port hardware init
05 028C 967 RSB : Continue PB search
028D 968 :
50 14 A5 3C 028D 969 120$: MOVZWL UCBS$L_FR4(R5),R0 : Set status info for SYSAP err routine
FD6C' 30 0291 970 BSBW SCSS$NOTIFY_SYSAP : Handle all CDT's in list
0294 971 :
05 0294 972 130$: RSB : Return
0295 973 :
0295 974 :
.DSABL LSB
```

```
0295 976 .SBTTL UNLOCK_BADQ, ZERO CORRUPTED QUEUE HDRS
0295 977
0295 978 :+
0295 979 : UNLOCK_BADQ tests the interlock bit on the queue pointed to by
0295 980 : R3. If the queue is locked, it is presumed corrupted and the header
0295 981 : zeroed so that any entries that should be in the queue are permanently
0295 982 : lost. The interlock bit should never be set in a power fail situation.
0295 983 : If it is, the auxiliary status in R1 is changed from SSS_POWERFAIL to
0295 984 : SSS_ABORT. If the queue is not locked, then return is taken without
0295 985 : altering anything -- the queue is purged later by ERR$CLEANUP_PKT.
0295 986
0295 987 Inputs:
0295 988
0295 989 R1 -Aux status to pass to SYSAPs
0295 990 R3 -Addr of queue header to check
0295 991
0295 992 Outputs:
0295 993
0295 994 All registers -Preserved
0295 995 :-
0295 996
0295 997 UNLOCK_BADQ:
0295 998
0295 999 BBC #0,(R3),Q_UNLOCKED ; Branch if queue unlocked
0364 8F 00 E1 0299 1000 CLRQ (R3) ; Else clear header
0364 8F 51 B1 029B 1001 CMPW R1,SS$ POWERFAIL ; Is this power failure?
0364 8F 03 12 02A0 1002 BNEQ Q_UNLOCKED ; Branch if not
0364 8F 51 2C 3C 02A2 1003 MOVZWL SS$ _ABORT,R1 ; Else change status to show
0364 8F 51 2C 3C 02A5 1004 ; SYSAPs that pkts are lost
0364 8F 51 2C 3C 02A5 1005
0364 8F 51 2C 3C 02A5 1006 Q_UNLOCKED:
0364 8F 51 2C 3C 02A5 1007
0364 8F 51 2C 3C 02A5 1008 RSB ; Return
0364 8F 51 2C 3C 02A6 1009
0364 8F 51 2C 3C 02A6 1010
0364 8F 51 2C 3C 02A6 1011 .DSABL LSB
```



```
02A6 1013      .SBTTL  ERR$DISC_PWFAIL,      PROCESS DISCONNECT CALL
02A6 1014      .SBTTL  -                      FOR CDT ON POWER
02A6 1015      .SBTTL  -                      FAILED PORT
02A6 1016
02A6 1017      :+
02A6 1018      ERR$DISC_PWFAIL is called by FPC$DCONNECT when the SYSAP issues
02A6 1019      a DISCONNECT for a connection associated with a power failed port.
02A6 1020      (Path block state = PB$C_PWR_FAIL.) In this case the local
02A6 1021      port is nonfunctional and action is to deallocate CDTs as they
02A6 1022      are DISCONNECTed after purging out the command queues of any SEND's
02A6 1023      the SYSAP may have done since being notified at its error entry.
02A6 1024
02A6 1025      If this is the last CDT on this path block, the path block (and
02A6 1026      system block) is removed and an attempt made to reinit the
02A6 1027      port hardware.
02A6 1028
02A6 1029      Inputs:
02A6 1030
02A6 1031      IPL                      -Fork IPL
02A6 1032
02A6 1033      R1                      -Addr of PB
02A6 1034      R3                      -Addr of CDT being DISCONNECTed
02A6 1035      R4                      -Addr of PDT
02A6 1036
02A6 1037      CDT$W_STATE             -Any except CLOSED or VC_FAIL
02A6 1038
02A6 1039      (SP)                   -Addr of return to FPC$DCONNECT
02A6 1040
02A6 1041      Outputs:
02A6 1042
02A6 1043      R0-R3                   -Destroyed
02A6 1044      Other registers       -Preserved
02A6 1045      :-
02A6 1046
02A6 1047      .ENABL  LSB
02A6 1048
02A6 1049      ERR$DISC_PWFAIL::
02A6 1050
28 51 DD 02A6 1051      PUSHL  R1                      : Save PB addr
A3 09 B1 02A8 1052      CMPW   CDT$W_STATE(R3),-      : Is this a listener with a
D9      02AB 1053      #CDT$C_CON_REC                : connect in hand?
05 12 02AC 1054      BNEQ    10$                      : Branch if not
FD4F' 30 02AE 1055      BSBW   SCSS$FREE_LISTEN        : Else just put it back to listening
11 11 02B1 1056      BRB     20$                      : Join common check for no more CDT's
02B3 1057
53 DD 02B3 1058 10$:  PUSHL  R3                      : Save CDT addr
0024 30 02B5 1059      BSBW   ERR$CLEANUP_PKT          : Purge out the command queues
02B8 1060      : again in case SYSAP error routine
02B8 1061      : did any more SENDs
53 BED0 02B8 1062      POPL   R3                      : Restore CDT addr
FD42' 30 02BB 1063      BSBW   SCSS$DEAL_SC$REC          : Deallocate CDT's SCS rcv buffer
00000000'GF 16 02BE 1064      JSB   G^SCSS$DEALL_CDT      : Deallocate CDT
02C4 1065
53 BED0 02C4 1066 20$:  POPL   R3                      : Retrieve PB addr in R3
34 A3 D5 02C7 1067      TSTL   PB$C_CDTLST(R3)          : Any CDT's left on PB?
0C 12 02CA 1068      BNEQ    30$                      : Branch if so
FD31' 30 02CC 1069      BSBW   CNF$REMOVE_PB            : Else deallocate PB/SB
```

PAERROR  
V04-001

Error Handling & Logging Routines I 14  
- FAILED PORT

16-SEP-1984 01:16:25 VAX/VMS Macro V04-00  
10-SEP-1984 01:16:10 [DRIVER.SRC]PAERROR.MAR;2

Page 24  
(14)

0112	C4	B5	02CF	1070	TSTW	PDT\$W_PBCOUNT(R4)	:	Any PB's left on this PDT?
	03	12	02D3	1071	BNEQ	30\$	:	Branch if some left
	007D	30	02D5	1072	BSBW	ERR\$INI PORT	:	Try to init port hardware now
			02D8	1073			:	
50	01	3C	02D8	1074	MOVZWL	#SS\$ _NORMAL, R0	:	Set to return success to SYSAP
		05	02DB	1075	RSB		:	
			02DC	1076			:	
			02DC	1077	.DSABL	LSB	:	

```
02DC 1079      .SBTTL  ERR$CLEANUP_PKT      CLEAN UP PACKETS QUEUED TO
02DC 1080      .SBTTL  -                    PORT AND IN LOGOUT AREA
02DC 1081
02DC 1082
02DC 1083      ;+
02DC 1084      ; ERR$CLEANUP_PKT calls FLUSH_Q to remove and dispose of packets currently
02DC 1085      ; on each of the port queues. It then extracts each packet address
02DC 1086      ; recorded in the logout area and calls ERR$DISP_ENTRY to dispose of the
02DC 1087      ; entry. The rule for disposing of packets is to return all packets
02DC 1088      ; to pool except send datagrams flagged as 'return to sysap.' These
02DC 1089      ; are returned to the SYSAP.
02DC 1090      Inputs:
02DC 1091
02DC 1092      R4                      -PDT addr
02DC 1093
02DC 1094      Outputs:
02DC 1095
02DC 1096      R0-R3                    -Destroyed
02DC 1097      :-
02DC 1098
02DC 1099      ASSUME  PDT$Q_COMQBASE  EQ  PDT$Q_COMQL
02DC 1100      ASSUME  PDT$Q_COMQL+8   EQ  PDT$Q_COMQH
02DC 1101      ASSUME  PDT$Q_COMQH+8   EQ  PDT$Q_COMQ2
02DC 1102      ASSUME  PDT$Q_COMQ2+8  EQ  PDT$Q_COMQ3
02DC 1103      ASSUME  PDT$Q_COMQ3+8  EQ  PDT$Q_RSPQ
02DC 1104
02DC 1105      .ENABL  LSB
02DC 1106
02DC 1107      ERR$CLEANUP_PKT::
02DC 1108
51  01E0 C4 DE 02DC 1109      MOVAL  PDT$Q_COMQBASE(R4),R1      ; Get adr of 1st command queue
53  53  05 D0 02E1 1110      MOVL   #<<PDT$Q_RSPQ - PDT$Q_COMQBASE>/8 + 1>,R3
02E4 1111      ; Get count of command/rsp queues
02E4 1112
02E4 1113      10$:  BSBB   FLUSH_Q      ; Purge next queue of all entries
51  51  08 C0 02E6 1114      ADDL   #8,R1      ; Step to next queue to flush
F8 53  F5 02E9 1115      SOBGTR  R3,10$      ; Branch if more queues
51  020C C4 D0 02EC 1116      MOVL   PDT$L_MFQHDR(R4),R1      ; Get addr of msg free queue header
020C 26 10 02F1 1117      BSBB   FLUSH_Q      ; Purge all entries
51  0208 C4 D0 02F3 1118      MOVL   PDT$L_DFQHDR(R4),R1      ; Get addr of dg free queue header
0208 1F 10 02F8 1119      BSBB   FLUSH_Q      ; Purge all entries
53  02E0 C4 DE 02FA 1120      MOVAL  PDT$L_DQEOLOGOUT(R4),R3      ; Get base of logout area
51  51  20 D0 02FF 1121      MOVL   #<<PDT$C_PALENGTH - PDT$L_DQEOLOGOUT>/4>,R1
0302 1122      ; Get count of elmts in logout area
0302 1123
0302 1124      20$:  MOVL   (R3)+,R2      ; Get addr of next entry
FFFFFFFF 8F 52 D1 0305 1125      CMPL   R2,#-1      ; Port record anything here?
07 13 030C 1126      BEQL   30$      ; Branch if not
0021 30 030E 1127      BSBW   ERR$DISP_ENTRY      ; Else dispose of entry
FC A3 00 D2 0311 1128      MCOML  #0,-4(R3)      ; Reset entry just processed
EA 51  F5 0315 1129      ;
0315 1130      30$:  SOBGTR  R1,20$      ; Branch if more entries in logout area
0318 1131      RSB      ; Return
0319 1132
0319 1133      .DSABL  LSB
0319 1134
0319 1135
```



```
0319 1137 .SBTTL FLUSH_Q REMOVE AND DISPOSE OF
0319 1138 .SBTTL - ALL QUEUED ENTRIES
0319 1139 .SBTTL ERR$DISP_ENTRY DISPOSE OF A SINGLE ENTRY
0319 1140
0319 1141
0319 1142 :+ FLUSH_Q removes and processes all entries from a specified port queue.
0319 1143
0319 1144 ERR$DISP_ENTRY processes a removed entry. All packets are returned to
0319 1145 pool except send datagrams flagged 'return to sysap.' These are
0319 1146 handled exactly as if they had gone out successfully.
0319 1147
0319 1148 Inputs:
0319 1149
0319 1150 IPL -Fork IPL
0319 1151 R1 -Addr of queue header (FLUSH_Q)
0319 1152 R2 -Pkt addr (ERR$DISP_ENTRY)
0319 1153
0319 1154 Outputs:
0319 1155
0319 1156 R0 -Destroyed
0319 1157 R2 -Destroyed (FLUSH_Q)
0319 1158 Other registers -Preserved
0319 1159
0319 1160
0319 1161 .ENABL LSB
0319 1162
0319 1163 FLUSH_Q:
0319 1164
0319 1165 $QRETRY REMQHI (R1),R2,- ; Remove next entry from
0319 1166 ERROR=FATALQ ; queue head
0319 1167 BVS 10$ ; Branch if no more entries
0319 1168 BSBB ERR$DISP_ENTRY ; Else dispose of entry
0319 1169 BRB FLUSH_Q ; Go for another entry
0319 1170
0319 1171 10$: RSB ; Return
0319 1172
0319 1173
0319 1174
0319 1175 ERR$DISP_ENTRY::
0319 1176
0319 1177 BBC #PPD$V_RSP,- ; Anybody expecting pkt?
0319 1178 PPD$B_FLAGS(R2),20$ ; Branch if not
0319 1179 CMPB PPD$B_OPC(R2),- ; Was it a send datagram?
0319 1180 #PPD$C_SNDDG
0319 1181 BNEQ 20$ ; Branch if not
0319 1182 PUSHF #^M<R1,R2,R3,R5> ; Save registers
0319 1183 BSBB INT$DISP_SENDDG ; Else handle as interrupt
0319 1184 POPR #^M<R1,R2,R3,R5> ; Restore destroyed registers
0319 1185 RSB ; Return
0319 1186
0319 1187 20$: BSBW INT$DEAL_PKT ; Return to pool
0319 1188 RSB ; Return
0319 1189
0319 1190 FATALQ: ; Should never get here since
0319 1191 ; queue lock cleared by UNLOCK_BADQ
0319 1192 BUGCHECK CIPORT, NONFATAL ; Nonfatal bugcheck
0319 1193
```

PAERROR  
V04-001

L 14

Error Handling & Logging Routines      16-SEP-1984 01:16:25 VAX/VMS Macro V04-00      Page 27  
ERR\$DISP\_ENTRY DISPOSE OF A SINGLE ENTR 10-SEP-1984 01:16:10 [DRIVER.SRC]PAERROR.MAR;2      (16)

61	7C	0350	1194	CLRQ	(R1)	:	If survive bugcheck, clear queue
		0352	1195			:	header
8E	D5	0352	1196	TSTL	(SP)+	:	Clear return from error call
	05	0354	1197	RSB		:	Return from FLUSH_Q
		0355	1198				
		0355	1199	.DSABL	LSB		

```
0355 1201      .SBTTL  ERR$INIPTS,          CALL PORT HARDWARE INIT
0355 1202
0355 1203      +
0355 1204      : If port has power now, call port initialization routine in PAINIT.
0355 1205      :
0355 1206      : Inputs:
0355 1207      :
0355 1208      : R4                      -PDT address
0355 1209      : IPL                  -IPL$_SCS
0355 1210      :
0355 1211      : Outputs:
0355 1212      :
0355 1213      : R0-R3                -Destroyed
0355 1214      : Other registers    -Preserved
0355 1215      : -
0355 1216
0355 1217      .DSABL  LSB
0355 1218
0355 1219  ERR$INIPTS::
0355 1220
0355 1221      BICW  #PDT$M PWF CLNUP,-      : Show power fail cleanup
0357 1222      PDT$W [PORT_STS(R4)         : bookkeeping done
035A 1223      BBC   #PDT$V PUP,-           : Has port got power now?
035C 1224      PDT$W [PORT_STS(R4),20$     : Branch if not
0360 1225      MOVQ  R4,-(SP)              : Save PDT addr and R5(
0363 1226      MOVL  PDT$L UCBO(R4),R5     : Get UCB addr for init
0368 1227      BBS   #UCB$V ONLINE,-      : If controller-unit already
036A 1228      UCBSW STS(R5),10$          : initialized, branch
036D 1229      MOVL  PDT$L CNF(R4),R4     : Get config register addr
0372 1230      MOVL  #PA PMC M MIN,-      : Place port in un-initialized state
0374 1231      PA PMC(R4)                : and disable device interrupts
0376 1232      BSBW  INT$PORT             : Call port initialization
0379 1233
0379 1234 10$:  MOVQ  (SP)+,R4             : Restore our registers
037C 1235
037C 1236 20$:  RSB                      : Return
037D 1237
037D 1238      .DSABL  LSB
```

01 AA 0355 1221  
0110 C4 0357 1222  
01 E1 035A 1223  
1C 0110 C4 035C 1224  
7E 54 7D 0360 1225  
55 00DC C4 D0 0363 1226  
04 E0 0368 1227  
0C 64 A5 036A 1228  
54 00E4 C4 D0 036D 1229  
01 D0 0372 1230  
04 A4 0374 1231  
FC87' 30 0376 1232  
54 8E 7D 0379 1233  
05 037C 1235  
037D 1237  
037D 1238



```
037D 1240 .SBTTL ERR$BUGCHECK, RECORD PORT LOCAL STORE
037D 1241 .SBTTL - IN MEMORY
037D 1242 .SBTTL ERR$BUGCHECKNF, RECORD LOCAL STORE CONDITIONALLY
037D 1243 .SBTTL - IF NONFATAL BUGCHECKS ARE FATAL
037D 1244
037D 1245 :+
037D 1246 : This routine copies the port local store (device registers, VC
037D 1247 : descriptor table, translation cache, work space, etc.) over the
037D 1248 : microcode in pool so that this info will be available in a dump.
037D 1249 :
037D 1250 : Inputs:
037D 1251 :
037D 1252 : R4 -PDT addr
037D 1253 :
037D 1254 : Outputs:
037D 1255 :
037D 1256 : All registers -Preserved
037D 1257 :-
037D 1258
037D 1259 ASSUME <^X1000> LE PA_C_WCSSIZ*6
037D 1260
037D 1261 .ENABL LSB
037D 1262
037D 1263 ERR$BUGCHECKNF::
037D 1264
00000000'8F E0 037D 1265 BBS #EXESV_FATAL_BUG,- : Branch if nonfatal bugchecks
00000000'GF 0383 1266 G^EXESGL_DEFFLAGS,- : are set to be fatal via
14 0388 1267 ERR$BUGCHECK : SYSGEN parameter
0389 1268 $DEBUGCHECK #ERRSV_DEB_BUGNF : If flag enabled, do a fatal bugcheck
039C 1269 : anyway regardless of SYSGEN param
05 039C 1270 RSB : Else return doing nothing
039D 1271
039D 1272 ERR$BUGCHECK::
039D 1273
039D 1274 PUSHF #^M<R0,R1,R2,R3> : Save caller's registers
52 00E4 C4 D0 039F 1275 MOVL PDT$L_CNF(R4),R2 : Get addr of base of device registers
03A4 1276
53 00000000'GF D0 03A4 1277 10$: MOVL G^SCS$GL_MCADR,R3 : Get addr of ucode in pool
51 0400 8F 3C 03AB 1278 MOVZWL #<^X100074>,R1 : Get size of device register space
03B0 1279
03B0 1280 20$: MOVL (R2)+,(R3)+ : Copy next long wd of local store
83 82 D0 03B0 1281 SOBGTR R1,20$ : Branch if more to copy
FA 51 F5 03B3 1281 POPR #^M<R0,R1,R2,R3>
03B6 1282
03B8 1283
05 03B8 1284 RSB
03B9 1285
03B9 1286 .DSABL LSB
```

```
03B9 1288 .SBTTL ERR$DEBUGCHECK, DEBUG BUGCHECK ENABLE FLAGS
03B9 1289
03B9 1290 ;+
03B9 1291 ; For the purpose of tracing intermittent problems that we normally
03B9 1292 ; attempt recovery from, a number of CIPORT bugchecks have been added.
03B9 1293 ; Each bugcheck is enabled or disabled by a separate flag. Flags
03B9 1294 ; may be turned on or off by a quick patch to location ERR$DEBUGCHECK.
03B9 1295 ; -
03B9 1296
03B9 1297 ;
03B9 1298 ; Flags are stored in the following longword:
03B9 1299 ;
03B9 1300
03B9 1301 ERR$DEBUGCHECK::
03B9 1302
00000000 03B9 1303 .LONG ^X0 ; The default is all bugchecks
03BD 1304 ; are disabled, and recovery enabled
03BD 1305
03BD 1306
03BD 1307 ;
03BD 1308 ; Flag definitions by bit number:
03BD 1309 ;
03BD 1310
00000000 03BD 1311 ERR$V_DEB_INVBN == 0 ; Invalid buffer name during blk
03BD 1312 ; xfer -- normally crashes port
03BD 1313
00000001 03BD 1314 ERR$V_DEB_BLV == 1 ; Local buffer length violation --
03BD 1315 ; normally crashes port
03BD 1316
00000002 03BD 1317 ERR$V_DEB_ACCV == 2 ; Access violation during blk
03BD 1318 ; xfer -- normally crashes port
03BD 1319
00000003 03BD 1320 ERR$V_DEB_PSV == 3 ; Packet size violation --
03BD 1321 ; normally crashes port
03BD 1322
00000005 03BD 1323 ERR$V_DEB_URP == 5 ; Unrecognized packet --
03BD 1324 ; normally logged and discarded
03BD 1325
00000006 03BD 1326 ERR$V_DEB_INVDP == 6 ; Invalid destination port number --
03BD 1327 ; normally crashes port
03BD 1328
00000007 03BD 1329 ERR$V_DEB_URC == 7 ; Unrecognized local command --
03BD 1330 ; normally crashes port
03BD 1331
00000008 03BD 1332 ERR$V_DEB_ABO == 8 ; Aborted command (port disabled) --
03BD 1333 ; normally crashes port
03BD 1334
00000009 03BD 1335 ERR$V_DEB_NPUPD == 9 ; No path + SNDMSG + unrecognized
03BD 1336 ; PPD type -- normally crashes port
03BD 1337
0000000A 03BD 1338 ERR$V_DEB_VCUPD ==10 ; VC closed + SNDMSG + unrecognized
03BD 1339 ; PPD type -- normally crashes port
03BD 1340
0000000B 03BD 1341 ERR$V_DEB_INVOP ==11 ; Invalid opcode in response --
03BD 1342 ; normally crashes port
03BD 1343
0000000C 03BD 1344 ERR$V_DEB_UNSTS ==12 ; Undefined status subtype in response --
```

	03BD	1345		: normally crashes port
	03BD	1346		
0000000D	03BD	1347	ERR\$V_DEB_NOSTS ==13	: Unrecognized combination of status,
	03BD	1348		: opcode, and PPD type --
	03BD	1349		: normally crashes port
0000000E	03BD	1350	ERR\$V_DEB_XCTER ==14	: XCT_ID sequence number check fails
	03BD	1351		: on DATREC/CNFREC. Normally crashes
	03BD	1352		: port
0000000F	03BD	1353	ERR\$V_DEB_SCERR ==15	: Source connection ID check fails --
	03BD	1354		: normally crashes port on MSGSNT
	03BD	1355		: and is ignored on DGSNT
00000010	03BD	1356	ERR\$V_DEB_NOPB ==16	: Rec'd connect request with no PB --
	03BD	1357		: normally crashes port
	03BD	1358		
00000011	03BD	1359	ERR\$V_DEB_CNFER ==17	: Entered VC cleanup with no PB --
	03BD	1360		: normally crashes port
	03BD	1361		
00000012	03BD	1362	ERR\$V_DEB_ILKQ ==18	: Interlock queue failure --
	03BD	1363		: normally crashes port
	03BD	1364		
00000013	03BD	1365	ERR\$V_DEB_NEPQ ==19	: Reiniting port with non empty
	03BD	1366		: command/response queues --
	03BD	1367		: normally logged and recovered
00000014	03BD	1368	ERR\$V_DEB_BUGNF ==20	: Nonfatal bugcheck being logged --
	03BD	1369		: normally continues
	03BD	1370		
00000015	03BD	1371	ERR\$V_DEB_PSRX ==21	: Undefined bits in PSR set --
	03BD	1372		: normally crashes port
	03BD	1373		
00000016	03BD	1374	ERR\$V_DEB_OSEQ ==22	: Port received response with
	03BD	1375		: sequence number mismatch. This
	03BD	1376		: is either a legitimate discard
	03BD	1377		: due to duplicate, or a sequence
	03BD	1378		: number error. Software normally
	03BD	1379		: crashes the vc.
	03BD	1380		
00000017	03BD	1381	ERR\$V_DEB_VCDCL ==23	: Port received sequenced message
	03BD	1382		: with VCD status set to closed.
	03BD	1383		: Software normally crashes the
	03BD	1384		: vc.
	03BD	1385		
00000018	03BD	1386	ERR\$V_DEB_MFQE ==24	: Port detected msg free queue
	03BD	1387		: empty.
	03BD	1388		: Normally, port crashes.
	03BD	1389		
	03BD	1390		



```
0380 1392      .SBTTL ELOG$INIT_SWERR,      LOG SOFTWARE ERROR
0380 1393      .SBTTL -                      ENCOUNTERED DURING
0380 1394      .SBTTL -                      PORT INITIALIZATION
0380 1395      .SBTTL ELOG$UCODE_NORD,      LOG MICROCODE NOT
0380 1396      .SBTTL -                      PROPERLY READ BACK
0380 1397      .SBTTL -                      ERROR
0380 1398      .SBTTL ELOG$HARDWARE,          LOG HARDWARE ERROR
0380 1399      .SBTTL ELOG$Q_INTRLOCK,        LOG QUEUE INTERLOCK
0380 1400      .SBTTL -                      FAILURE
```

```
0380 1401
0380 1402
0380 1403 :+ These routines log those errors which use the device attention, EMBSC_DA,
0380 1404 : error-log-entry format. There are three such error types:
0380 1405 :   - Software errors detected during port initialization.
0380 1406 :   - Microcode failed to read-back as loaded (this is logged as a special
0380 1407 :     type hardware error).
0380 1408 :   - CPU or port ucode not at adequate rev level.
0380 1409 :   - Hardware error (typical to but more extensive than those found
0380 1410 :     in more standard I/O devices).
0380 1411 :   - Failures to obtain access to a queue because of its interlock.
```

```
0380 1412
0380 1413 : After some entry specific processing, the body of this routine calls OPA0 LOG
0380 1414 : to broadcast the error to OPA0, if indicated, and then uses ERL$DEVICEATTN
0380 1415 : to log the error. ERL$DEVICEATTN will call ELOG$REGDUMP which will actually
0380 1416 : copy the appropriate information into the error log.
```

```
0380 1417
0380 1418
```

```
0380 1419 : ELOG$INIT_SWERR:
```

```
0380 1420
```

```
0380 1421 : Inputs:
```

```
0380 1422 :   R0
```

```
0380 1423 :   Sign bit set indicates that the error will crash port
```

```
0380 1424 :   Sign bit not set indicates that it will not
```

```
0380 1425 :   R5
```

```
0380 1426 :   - Address of device UCB
```

```
0380 1427
```

```
0380 1428 : ELOG$UCODE_NORD:
```

```
0380 1429
```

```
0380 1430 : Inputs:
```

```
0380 1431 :   R0
```

```
0380 1432 :   R4
```

```
0380 1433 :   R5
```

```
0380 1434 :   - Correct microcode value
```

```
0380 1435 :   - Base virtual address of CI port registers
```

```
0380 1436 :   - Address of device UCB
```

```
0380 1437
```

```
0380 1438 : ELOG$CPU_REV:
```

```
0380 1439
```

```
0380 1440 : Inputs:
```

```
0380 1441 :   R1
```

```
0380 1442 :   R5
```

```
0380 1443 :   -System ID Register which contains CPU rev level
```

```
0380 1444 :   -UCB addr
```

```
0380 1445
```

```
0380 1446 : ELOG$UCODE_ERR, ELOG$UCODE_WARN:
```

```
0380 1447
```

```
0380 1448 : Inputs:
```

```
0380 1449 :   R2
```

```
0380 1450 :   R5
```

```
0380 1451 :   -Addr of IDREC pkt containing port ucode rev
```

```
0380 1452 :   level at offset PPD$L_RPORT_REV
```

```
0380 1453 :   -UCB addr
```

```
0380 1454 :
```

```
038D 1449 : ELOG$HARDWARE:
038D 1450 :
038D 1451 : Inputs:
038D 1452 :     R0      - Error subtype code in bits 0 through 7
038D 1453 :             Sign bit set indicates that the error will crash port
038D 1454 :             Sign bit not set indicates that it will not
038D 1455 :     R4      - Base virtual address of CI port registers
038D 1456 :     R5      - Address of device UCB
038D 1457 :
038D 1458 :
038D 1459 : ELOG$Q_INTRLOCK:
038D 1460 :
038D 1461 : Inputs:
038D 1462 :     R0      - Error subtype code in bits 0 through 7
038D 1463 :             Sign bit set indicates that the error will crash port
038D 1464 :             Sign bit not set indicates that it will not
038D 1465 :     R4      - Address of PDT
038D 1466 :
038D 1467 :
038D 1468 : ALL ROUTINES:
038D 1469 :
038D 1470 : Outputs:
038D 1471 :     R0 is destroyed. All other registers are preserved. An entry is made
038D 1472 :     in the error log. The existence of this error might have been broadcast
038D 1473 :     to _OPA0.
038D 1474 :
038D 1475 :
038D 1476 : SPECIAL NOTES:
038D 1477 :
038D 1478 : Proper operation of this routine, and ELOG$REGDUMP, depends upon
038D 1479 : ERL$DEVICEATTN passing R4 and R5 unaltered to ELOG$REGDUMP. As of this
038D 1480 : routines writing, this was the case.
038D 1481 : -
038D 1482 :
038D 1483 : +
038D 1484 : The following are various values related to or controlling the size of a
038D 1485 : device attention error log entry for this device.
038D 1486 : -
038D 1487 :
00000006 038D 1488 PORT_REGS_LOGGED = 6      : Number of port registers logged
00000003 038D 1489 NUM_EX_LONGWORDS = 3    : Number of extra longwords logged
038D 1490 TOTAL_LONGWORDS = 2 -          : Longword count + error type code
038D 1491      + PORT_REGS_LOGGED -          : + port registers
00000008 038D 1492      + NUM_EX_LONGWORDS          : + extra longwords
038D 1493 :
038D 1494 ELOG$K_BYTES == <TOTAL_LONGWORDS * 4> - : This is the number of bytes in a
0000007A 038D 1495      + EMB$C_DV_REGSAV              : device attention error log entry
038D 1496 :                                         : from the CI as entered in the DDI.
038D 1497 :
038D 1498 .MACRO ZERO_EXTRA_LONGWORDS
038D 1499 ASSUME NUM_EX_LONGWORDS EQ 3
038D 1500 CLRQ  -(SP)
038D 1501 CLRL  -(SP)
038D 1502 .ENDM ZERO_EXTRA_LONGWORDS
038D 1503 :
0000003E 038D 1504 DA_MASK = ^M<R1,R2,R3,R4,R5>
038D 1505 :
```

```
03BD 1506 ELOG$INIT_SWERR:: ; Software error during initialization
03BD 1507
3E BB 03BD 1508 PUSHR #DA_MASK ; Save registers.
54 D4 03BF 1509 CLRL R4 ; Zero port base VA implying don't log
; port registers.
03C1 1510 ZERO EXTRA LONGWORDS ; No extra longword to log here.
03C1 1511
03C5 1512 ASSUME PAERSK_ET_INSW EQ 0
7E 94 03C5 1513 CLRB -(SP) ; Build error type part of error code.
72 11 03C7 1514 BRB ELOG$LOG_DA ; Branch to common code.
03C9 1515
03C9 1516
03C9 1517 ELOG$UCODE_NORD::
03C9 1518
3E BB 03C9 1519 PUSHR #DA_MASK ; Save registers.
03CB 1520 ASSUME NUM_EX_LONGWORDS EQ 3
50 DD 03CB 1521 PUSHL R0 ; Ex. lw. 3 = correct ucode value.
7E 7C 03CD 1522 CLRL -(SP) ; Init ex. lw. 1 & 2 to zero.
55 SE DO 03CF 1523 MOVL SP, R5 ; Save current stack pointer.
03D2 1524 $PRTCTINI - ; Protect the following device register
03D2 1525 B*108, #MCHKSM NEXM ; references from machine checks.
04 A5 18 A4 DO 03DE 1526 MOVL PA_MDATA(R4), 4(R5) ; Ex. lw. 2 = wrong ucode value.
65 14 A4 DO 03E3 1527 MOVL PA_MADR(R4), (R5) ; Ex. lw. 1 = failing ucode address.
03E7 1528 $PRTCTEND TOS ; If check occurs, leave zero values(s).
55 40 AE DO 03E8 1529 MOVL 3*4+4*4(SP), R5 ; Restore previously saved UCB addr.
50 8000 8F 32 03EC 1530 CVTWL #<PAERSK_ES_UCDW ! ^X8000>, - ; Plant error subtype
03F1 1531 R0 ; w/ crash port code.
2D 11 03F1 1532 BRB LOG_AS_HARDWARE ; Branch to common hardware error
03F3 1533 ; logging code.
03F3 1534
03F3 1535 ELOG$CPU_REV::
03F3 1536
3E BB 03F3 1537 PUSHR #DA_MASK ; Save registers
03F5 1538 ASSUME NUM_EX_LONGWORDS EQ 3
51 DD 03F5 1539 PUSHL R1 ; 1st extra longwd gets CPU SID
8007 8F 32 03F7 1540 CVTWL #<PAERSK_ES_CPUREV ! ^X8000>, -
50 03FB 1541 R0 ; Set error subtype, port shutting down
16 11 03FC 1542 BRB REV_ERROR ; Join common rev error logging
03FE 1543
03FE 1544
03FE 1545 ELOG$UCODE_ERR::
03FE 1546
8006 3E BB 03FE 1547 PUSHR #DA_MASK ; Save registers
8F 32 0400 1548 CVTWL #<PAERSK_ES_REVER ! ^X8000>, -
50 0404 1549 R0 ; Set error subtype, port shuts down
05 11 0405 1550 BRB PORT_UCODE ; Join common port rev error logging
0407 1551
0407 1552
0407 1553 ELOG$UCODE_WARN::
0407 1554
50 3E BB 0407 1555 PUSHR #DA_MASK ; Save registers
08 9A 0409 1556 MOVZBL #PAERSK_ES_REVCA, R0 ; Set error subtype, non fatal to port
040C 1557
040C 1558 PORT_UCODE:
040C 1559
040C 1560 ASSUME NUM_EX_LONGWORDS EQ 3
1C A2 DD 040C 1561 PUSHL PPD$L RPORT_REV(R2) ; 1st extra longwd gets port rev level
00B8 C5 6E DO 040F 1562 MOVL (SP), DCB$T_OPA0_TEMP(R5) ; Save rev level to format in opa0 msg
```



```
0414 1563
0414 1564 REV_ERROR:
0414 1565
7E 7C 0414 1566 CLRQ -(SP) ; 2nd and 3rd longwds not used
54 D4 0416 1567 CLRL R4 ; Zero port CNF addr to avoid logging
; device registers
06 11 0418 1568 BRB LOG_AS_HARDWARE ; Join common HW type error logging
041A 1570
041A 1571 ELOG$HARDWARE::
041A 1572
3E BB 041A 1573 PUSHR #DA_MASK ; Save registers.
041C 1574 ZERO_EXTRA_LONGWORDS ; No extra longword to log here.
0420 1575 LOG_AS_HARDWARE:
7E 01 90 0420 1576 MOVB #PAERSK_ET_HW, -(SP) ; Build error type part of error code.
16 11 0423 1577 BRB ELOG$$LOG_DA ; Branch to common code.
0425 1578
0425 1579
0425 1580 ELOG$INTRLOCK::
0425 1581
0425 1582
55 00DC 3E BB 0425 1582 PUSHR #DA_MASK ; Save registers.
54 24 A5 D0 0427 1583 MOVL PDT$L_UCB0(R4), R5 ; Obtain UCB address.
54 2C B4 D0 042C 1584 MOVL UCBS$L_CRB(R5), R4 ; Get base VA of port registers via
; UCB ==> CRB ==> IDB ==> CSR.
0430 1585 ASSUME IDBS$L_CSR EQ 0
0430 1586 MOVL @CRBS$L_INTD+VECS$L_IDB(R4), R4
0434 1587 ZERO_EXTRA_LONGWORDS ; No extra longword to log here.
7E 02 90 0438 1588 MOVB #PAERSK_ET_ILCK, -(SP) ; Build error type part of error code.
043B 1589 BRB ELOG$$LOG_DA ; Branch to common code.
043B 1590
00000014 043B 1591 CLN_BYTES = <NUM_EX_LONGWORDS * 4> + 8 ; Number of bytes to clean from stack
043B 1592
043B 1593 ELOG$$LOG_DA:
043B 1594
50 D5 043B 1595 TSTL R0 ; Is the port going to be crashed?
04 18 043D 1596 BGEQ 10$ ; Branch if no. Otherwise,
6E 80 8F 88 043F 1597 BISB #PAERSM_CPRT, (SP) ; set the right bit in error code.
7E 50 90 0443 1598 MOVB R0, -(SP) ; Add error subtype to error code.
7E B4 0446 1599 CLRW -(SP) ; Longword align the stack.
7E 54 D0 0448 1600 MOVL R4, -(SP) ; Save VA of port registers.
044B 1601
044B 1602
50 D4 044B 1602 CLRL R0 ; Clear register
50 06 AE 8000 8F AB 044D 1603 BICW3 #X8000,6(SP),R0 ; Retrieve error subtype and type
51 FBB8 CF 9E 0454 1604 MOVAB DA_OPA0_LOG_TAB,R1 ; Retrieve device attention_OPA0 table
53 55 D0 0459 1605 MOVL R5,R3 ; Move UCB address into proper register
023B 30 045C 1606 BSBW OPA0_LOG ; Broadcast error to_OPA0 if indicated
55 000000A0 8F C2 045F 1607 SUBL2 #UCBS$L_MSGFKBLK,R5 ; Compute UCB address
0466 1608
54 5E D0 0466 1609 MOVL SP, R4 ; Set pointer needed by ELOG$REGDUMP.
00000000 GF 16 0469 1610 JSB G^ERL$DEVICEATTN ; Perform actual error logging.
5E 14 AE 9E 046F 1611 MOVAB CLN_BYTES(SP), SP ; Clean saved information from stack.
3E BA 0473 1612 POPR #DA_MASK ; Restore saved registers
05 0475 1613 RSB ; Return to caller.
```

```
0476 1615 .SBTTL ELOG$REGDUMP, DEVICE ATTENTION
0476 1616 .SBTTL - REGISTER DUMP ROUTINE
0476 1617
0476 1618 :+
0476 1619 : This routine is called by ERL$DEVICEATTN (which is called by ELOG$$LOG_DA)
0476 1620 : to copy the appropriate device registers into an error log entry.
0476 1621
0476 1622 : Inputs:
0476 1623 : R0 - Starting address in error log buffer to be filled
0476 1624 : 00(R4) - Base virtual address of CI port registers
0476 1625 : 04(R4) - filler word of zeros
0476 1626 : 06(R4) - Error code type, crash port, subtype fields
0476 1627 : 08(R4) - NUM_EX_LONGWORDS of additional data to be logged
0476 1628 : R5 - Address of the device UCB
0476 1629
0476 1630 : Outputs:
0476 1631 : CI port register values and the additional data are copied to the
0476 1632 : location(s) pointed to by R0. R0, R1 and R2 are destroyed. If for
0476 1633 : any reason the CI port registers are inaccessible, zeros will be
0476 1634 : logged for thier values.
0476 1635 :-
0476 1636
0476 1637
0476 1638 ELOG$REGDUMP::
0476 1639
0476 1640 52 50 D0 0476 1640 MOVL R0, R2 ; Copy buffer address to a safe place.
0476 1641 82 0A 9A 0479 1641 MOVZBL #<TOTAL_LONGWORDS - 1>, - ; Insert count of longword
0476 1642 047C 1642 (R2)+ ; "registers" to be logged.
0476 1643 82 06 A4 B0 047C 1643 MOVW 6(R4), (R2)+ ; Insert error retry counts, type,
0476 1644 82 0080 C5 B0 0480 1644 MOVW UCBSB_ERTCNT(R5), (R2)+ ; subtype, and crash port information
0476 1645 0485 1645 ASSUME PORT_REGS_LOGGED EQ 6 ; to form PADRIVER error code.
0476 1646 0485 1646 CLRQ (R2) ; Zero places where CI port registers
0476 1647 08 7C 0487 1647 CLRQ 8(R2) ; may be copied.
0476 1648 10 A_ 7C 048A 1648 CLRQ 16(R2)
0476 1649 048D 1649 ASSUME NUM_EX_LONGWORDS EQ 3
0476 1650 18 A2 08 A4 7D 048D 1650 MOVQ 8(R4), 24(R2) ; Copy extra longwords into
0476 1651 20 A2 10 A4 D0 0492 1651 MOVL 16(R4), 32(R2) ; into error log entry.
0476 1652 51 64 D0 0497 1652 MOVL (R4), R1 ; Obtain base VA of CI port registers.
0476 1653 049A 1653 BEQL 100$ ; If zero, don't log registers.
0476 1654 049C 1654 $PRCTINI - ; Protect the following device register
0476 1655 049C 1655 B*10$, MCHK$M_NEXM ; references from machine checks.
0476 1656 04AC 1656 MOVL PA_CNF(R1), (R2) ; Plant configuration register.
0476 1657 04AF 1657 MOVL PA_PMC(R1), 4(R2) ; Plant maintenance control/status reg.
0476 1658 08 A2 0900 C1 D0 04B4 1658 MOVL PA_PS(R1), 8(R2) ; Plant port status register.
0476 1659 0C A2 0938 C1 D0 04BA 1659 MOVL PA_PPAR(R1), 12(R2) ; Plant failing address register.
0476 1660 10 A2 093C C1 D0 04C0 1660 MOVL PA_PESR(R1), 16(R2) ; Plant port error status register.
0476 1661 14 A2 0940 C1 D0 04C6 1661 MOVL PA_PPR(R1), 20(R2) ; Plant port parameter register.
0476 1662 04CC 1662 $PRCTEND TOS ; End protected code.
0476 1663 05 04CD 1663 100$: RSB ; Return to ERL$DEVICEATTN.
```

04CE 1665	.SBTTL	ELOG\$PACKET,	LOG PACKET RELATED
04CE 1666	.SBTTL	-	ERROR, GENERAL CASE
04CE 1667	.SBTTL	ELOG\$CABLES,	LOG CABLE STATUS
04CE 1668	.SBTTL	-	CHANGE, GENERAL CASE
04CE 1669	.SBTTL	ELOG\$PTH_ST_CHG	LOG PATH STATUS
04CE 1670	.SBTTL	-	CHANGE
04CE 1671	.SBTTL	ELOG\$CBL_X_CHG	LOG CABLES CROSSED OR
04CE 1672	.SBTTL	-	NOT CROSSED STATUS
04CE 1673	.SBTTL	-	CHANGE
04CE 1674	.SBTTL	ELOG\$ERROR_DG	LOG ERROR LOG DATAGRAM

04CE 1676 :+ These routines log those errors which use the logged message, EMBSC\_LM, error-log-entry format. All such errors result from detection of an exceptional condition in a data packet. The error log entry produced by these routines will include upto 72 bytes of the packet which signaled the exceptional condition starting with the 12th byte of the packet.

04CE 1683 : There is one exceptional case, and that is when what is being logged is the refusal of the local system to open up a virtual circuit to a remote system because the information provided by the remote system conflicts with information that is already present within the system-wide configuration data base. In such a case what is logged instead of a data packet is the remote system node name, the known system nodename, and the known system ID.

04CE 1690 : Before calling ERL\$LOGMESSAGE to log the error condition, these routines call OPA0\_LOG to log the condition to \_OPA0, if such a broadcast is warranted.

04CE 1693 : As a matter of convenience, there are four entry points to the routine, one for each of the following conditions:

- 04CE 1695 : - A path status change (good to bad, or bad to good)
  - 04CE 1696 : - A cables crossed/uncrossed status change
  - 04CE 1697 : - All other errors detected with in a packet
  - 04CE 1698 : - An error log datagram, specified by the PPD type = 5 (PPD\$C\_ELOG)
- 04CE 1699 : These are used for sending an error log message to a system without necessarily having a connection to the system over which to send error log info.

04CE 1702 : ELOG\$PTH\_ST\_CHG:

04CE 1703 : Inputs:

04CE 1704 : R0

- 04CE 1706 : - Address of previous path status information byte.

04CE 1707 : In this byte:

04CE 1708 : PB\$M\_CUR\_PS eq 0 ==> path was broken

04CE 1709 : PB\$M\_CUR\_PS ne 0 ==> path was good

04CE 1710 : The address is assumed to be one of PB\$B\_PO\_STS(R1) or PB\$B\_P1\_STS(R1). This information is used to determine which path is being described.

04CE 1711 : R1

04CE 1712 : R2

04CE 1713 : R4

04CE 1714 : - PB address

04CE 1715 : - Packet address

04CE 1716 : - PDT address

04CE 1717 : ELOG\$CBL\_X\_CHG:

04CE 1718 : Inputs:

04CE 1719 : R1

- 04CE 1720 : - 0 ==> cables currently crossed



```
04CE 1722 : 1 ==> cables currently uncrossed
04CE 1723 : R2 - Packet address
04CE 1724 : R3 - PB address
04CE 1725 : R4 - PDT address
04CE 1726 :
04CE 1727 :
04CE 1728 : ELOG$PACKET: and ELOG$CABLES:
04CE 1729 :
04CE 1730 : Inputs:
04CE 1731 : R0 - Error subtype code in bits 0 through 7
04CE 1732 : Sign bit set indicates that the error will crash port
04CE 1733 : Sign bit not set indicates that it will not
04CE 1734 : R1 - PB address (ELOG$PACKET only)
04CE 1735 : R2 - Packet address (zero if none exists)
04CE 1736 : R4 - PDT address
04CE 1737 : R5 - Known system SB address
04CE 1738 : (ELOG$PACKET and subtype = PAER$K_ES_RSCKS only)
04CE 1739 :
04CE 1740 : ELOG$ERROR_DG:
04CE 1741 :
04CE 1742 : Inputs:
04CE 1743 : R2 -Error log packet address
04CE 1744 : R3 -PB address
04CE 1745 : R4 -PDT address
04CE 1746 :
04CE 1747 : ALL ROUTINES:
04CE 1748 :
04CE 1749 : Outputs:
04CE 1750 : All other registers are preserved. An entry is made in the error log.
04CE 1751 : The existence of this error might have been broadcast to _OPA0.
04CE 1752 : -
04CE 1753 :
0000003F 04CE 1754 LM MASK = ^M<R0,R1,R2,R3,R4,R5>
00000014 04CE 1755 SAVEDR5 = 4*5
04CE 1756 :
04CE 1757 ELOG$PTH_ST_CHG:: : Path status change
04CE 1758 :
04CE 1759 PUSH R0,R1,R2,R3,R4,R5 : Save registers.
04D0 1760 ASSUME PAER$K_ES_OGB EQ 0
04D0 1761 CLRL R5 : Assume it went from good to bad.
04D2 1762 BLBS (R0), 10$ : Branch if old status was good.
04D5 1763 MOVZBL #PAER$K_ES_OBG, R5 : Else, it went from bad to good.
04D8 1764 10$: : Determine which path was effected by subtracting the address of the
04D8 1765 : path 0 status byte from the address of the status byte passed to us.
04D8 1766 : Then add the good-to-bad or bad-to-good subtype code base to form
04D8 1767 : the error subtype code.
04D8 1768 ASSUME PB$B P1 STS EQ PB$B P0 STS+1
04D8 1769 ASSUME PAER$K_ES_1GB EQ PAER$K_ES_OGB+1
04D8 1770 ASSUME PAER$K_ES_1BG EQ PAER$K_ES_OBG+1
04D8 1771 MOVAB PB$B P0 STS(R1), R3 : Get path 0 status byte address.
04DC 1772 SUBL R3, R0 : Subtract from passed address.
04DF 1773 ADDL R5, R0 : Add error subtype code base.
04E2 1774 BRB LOG_AS_CHANGE : Branch to common state change code.
04E4 1775 :
04E4 1776 :
04E4 1777 ELOG$CBL_X_CHG:: : Cables crossed/uncrossed change
04E4 1778 :
```

```

      3F  BB  04E4 1779      PUSH  #LM MASK      ; Save registers.
      04E6 1780      ASSUME PAERSK_ES_CU EQ PAERSK_ES_UC+1
      04E6 1781      ASSUME PB$M COR CBL EQ 1
50    51  04  C1  04E6 1782      ADDL3 #PAERSK_ES_UC, R1, R0 ; Form change crossing subtype.
      51  53  D0  04EA 1783      MOVL  R3, R1 ; Move PB address to right place.
      04ED 1784 LOG_AS_CHANGE:
55    41  8F  9A  04ED 1785      MOVZBL #PAERSK_ET_CBL, R5 ; Set cable status change error type.
      33  11  04F1 1786      BRB  ELOG$$LOG_CM ; Branch to common code.
      04F3 1787
      04F3 1788
      04F3 1789 ELOG$CABLES:: ; Cables change of state, general case
      04F3 1790
      04F3 1791      .ENABL LSB
      04F3 1792
      3F  BB  04F3 1793      PUSH  #LM MASK      ; Save registers.
55    41  8F  9A  04F5 1794      MOVZBL #PAERSK_ET_CBL, R5 ; Set cable status change error type.
      51  D4  04F9 1795      CLRL  R1 ; Assume no PB
      52  D5  04FB 1796      TSTL  R2 ; Is there a message?
      14  13  04FD 1797      BEQL  10$ ; Branch if no message.
      FAFE' 30  04FF 1798      BSBW  CNF$LKP_PB_MSG ; Attempt to find path block.
      0F  11  0502 1799      BRB  10$ ; Join common code
      0504 1800
      0504 1801
      0504 1802 ELOG$PACKET:: ; Packet error, general case
      0504 1803
      3F  BB  0504 1804      PUSH  #LM MASK      ; Save registers.
52    00B4 C4  C2  0506 1805      SUBL  PDT$L_MSGHDRSZ(R4), R2 ; Back the pointer up
      02  11  0508 1806      BRB  5$
      050D 1807
      050D 1808 ELOG$PACKET1:: ; Packet error, general case
      050D 1809
      3F  BB  050D 1810      PUSH  #LM MASK      ; Save registers.
55    40  8F  9A  050F 1811 5$: MOVZBL #PAERSK_ET_PKT, R5 ; Set packet error type.
      50  6E  D0  0513 1812 10$: MOVL  (SP), R0 ; Restore caller's error subtype.
      0E  11  0516 1813      BRB  ELOG$$LOG_CM ; Go to common code.
      0518 1814
      0518 1815      .DSABL LSB
      0518 1816
      0518 1817
      0518 1818 ELOG$ERROR_DG:: ; Error log datagram to log
      0518 1819
      3F  BB  0518 1820      PUSH  #LM MASK      ; Save registers
50    07  9A  051A 1821      MOVZBL #PAERSK_ES_ERRDG, R0 ; Get error subtype
      51  53  D0  051D 1822      MOVL  R3, R1 ; Copy PB address
      55  40  8F  9A  0520 1823      MOVZBL #PAERSK_ET_PKT, R5 ; Get error type
      00  11  0524 1824      BRB  ELOG$$LOG_CM ; Join common code to set up
      0526 1825 ; error log entry and log it
```

```
0526 1827 :+
0526 1828 : At this point the registers have the following values:
0526 1829 :
0526 1830 : R0 - Error subtype code in bits 0 through 7
0526 1831 : Sign bit set indicates that the error will crash port
0526 1832 : Sign bit not set indicates that it will not
0526 1833 : R1 - =0 ==> no PB exists
0526 1834 : Otherwise R1 = PB address
0526 1835 : R2 - Packet address (zero if none exists)
0526 1836 : R4 - PDT address
0526 1837 : R5 - Error type code
0526 1838 :
0526 1839 : The following code will build the logged message buffer in a UCB extension,
0526 1840 : and cause it to be placed in the error log. It will also call OPA0_LOG
0526 1841 : to broadcast the error to OPA0 if such a broadcast is required.
0526 1842 : Synchronization on use of the UCB extension area for this purpose is
0526 1843 : accomplished via the UCBSM_ERLOGIP bit in UCBSW_STS.
0526 1844 :
0526 1845 : Because some of the entities in a logged message have odd sizes, the
0526 1846 : following code sometimes saves instructions by incorrectly writing longer
0526 1847 : than necessary entities, and later overwriting the high order portions of
0526 1848 : the written data with the correct information.
0526 1849 :-
0526 1850 :
0526 1851 :
0526 1852 ELOG$$LOG_LM:
0526 1853 :
0526 1854 : MOVL PDT$UCB(R4), R3 ; Get the UCB address.
0526 1855 : BBSC #UCBS$ERLOGIP, - ; Flag error logging in progress and
0526 1856 : UCBSW_STS(R3), 5$ ; branch if none previously in progress.
0526 1857 : BRW 90$ ; Branch if error log is in progress.
0526 1858 5$: MOVB R0, UCBSB_LMEST(R3) ; Plant error subtype value.
0526 1859 : MOVB R5, UCBSB_LMET(R3) ; Plant error type value.
0526 1860 : TSTL R0 ; Is the port going to be crashed?
0526 1861 : BGEQ 10$ ; Branch if no. Otherwise, set flag
0526 1862 : BISB #PAERSM CPRT, UCBSB_LMET(R3) ; bit in error code byte.
0526 1863 10$: MOVW UCBSB_ERTCNT(R3), - ; Plant error retry and max retry
0526 1864 : UCBSB_LMERTCNT(R3) ; counts.
0526 1865 : ADDW3 #1, UCBSW_ERRCNT(R3), R0 ; Adjust unincremented error counter,
0526 1866 : MOVZWL R0, UCBSW_LMERRCNT(R3) ; plant it, and zero word following it.
0526 1867 : ASSUME UCBS$LSADDR EQ 6
0526 1868 : ASSUME UCBS$LSID EQ 6
0526 1869 : ASSUME UCBS$RSADDR EQ 6
0526 1870 : ASSUME UCBS$RSID EQ 6
0526 1871 : ASSUME SB$SYSTEMID EQ 6
0526 1872 : $PRTCTINI - ; Protect the following device register
0526 1873 : B*20$, MCHK$NEXM ; reference from machine checks.
0526 1874 : MOVL @PDT$L_PPR(R4), - ; Get the local station address
0526 1875 : UCBSN_LSADDR(R3) ; directly from the port.
0526 1876 : $PRTCTEND 20$ ; End protected code.
0526 1877 : BLBS R0, 25$ ; Branch if no machine check occurred.
0526 1878 : MNEGL #1, UCBSN_LSADDR(R3) ; If couldn't get local station
0526 1879 : MNEGW #1, UCBSN_LSADDR+4(R3) ; address, put all ones in its place.
0526 1880 : BRB 30$ ; Then, continue with processing.
0526 1881 25$: CLRL UCBSN_LSADDR+2(R3) ; If got address, clear high order bits.
0526 1882 30$: MOVL G*SCS$GB_SYSTEMID, - ; Get local system id from system
0526 1883 : UCBSN_LSID(R3) ; global address.
```



```
00000004'GF B0 058D 1884 MOVW G*SCS$GB_SYSTEMID+4,- : Copy h.o. 2 bytes of system id
00E2 C3 0593 1885 UCB$N_LSID+4(R3)
00EA C3 7C 0596 1886 CLRQ UCB$N_RSID(R3) : Assume remote system id won't be
059A 1887 : found and zero it (plus a little).
059A 1888
059A 1889 ASSUME UCB$N_RSADDR+6 EQ UCB$N_RSID
059A 1890 ASSUME UCB$N_RSID+6 EQ UCB$N_CICMD
059A 1891 ASSUME SB$S_NODENAME EQ 16
059A 1892
00D0 C3 B1 059A 1893 CMPW UCB$B_LMEST(R3),- : Logging known-remote system conflict?
4008 8F 059E 1894 #<PAER$K_ET_PKT$B + PAER$K_ES_RSCK$>
3D 12 05A1 1895 BNEQ 32$ : Branch if not
55 14 AE D0 05A3 1896 MOVL SAVEDR5(SP),R5 : Otherwise restore known system SB addr
52 30 A1 D0 05A7 1897 MOVL PB$S_SBLINK(R1),R2 : Retrieve remote system SB address
50 00E4 C3 9E 05AB 1898 MOVAB UCB$N_RSADDR(R3),R0 : Position to remote system address
05B0 1899 : field within logged msg working buffer
80 0C A1 D0 05B0 1900 MOVL PB$B_RSTATION(R1),(R0)+ : Store remote station address
80 80 B4 05B4 1901 CLRW (R0)+
80 18 A2 D0 05B6 1902 MOVL SB$B_SYSTEMID(R2),(R0)+ : Store remote system ID
80 1C A2 B0 05BA 1903 MOVW SB$B_SYSTEMID+4(R2),(R0)+
80 18 A5 D0 05BE 1904 MOVL SB$B_SYSTEMID(R5),(R0)+ : Store known system ID
80 1C A5 B0 05C2 1905 MOVW SB$B_SYSTEMID+4(R5),(R0)+
80 44 A5 7D 05C6 1906 MOVQ SB$T_NODENAME(R5),(R0)+ : Store known system nodename
80 4C A5 7D 05CA 1907 MOVQ SB$T_NODENAME+8(R5),(R0)+
80 44 A2 7D 05CE 1908 MOVQ SB$T_NODENAME(R2),(R0)+ : Store remote system nodename
80 4C A2 7D 05D2 1909 MOVQ SB$T_NODENAME+8(R2),(R0)+
00 63 00 DD 05D6 1910 PUSHL R3 : Save UCB address
60 22 2C 05D8 1911 MOVCS #0,(R3),#0,- : Clear remainder of logged msg buffer
5D 11 05DE 1912 #<UCB$K_LMPKTBYTS-30>,(R0)
05E0 1913 BRB 66$ : Go finish logged message
05E0 1914
00E4 C3 52 D5 05E0 1915 32$: TSTL R2 : Is there a message packet?
00E8 C3 18 12 05E2 1916 BNEQ 35$ : Branch if there is one.
05E4 1917 MNEGL #1, UCB$N_RSADDR(R3) : Else, can't get remote station
05E9 1918 MNEGL #1, UCB$N_RSADDR+4(R3) : address, so put all ones in its place.
05EE 1919 PUSHL R3 : Save UCB address.
004B 8F 00 63 00 2C 05F0 1920 MOVCS #0,(R3),#0,- : Zero all of logged message buffer
00F0 C3 05F7
05FA 1921 #<UCB$K_LMPKTBYTS+8>,- : in which message packet would
05FA 1922 UCB$N_CICMD(R3) : normally be put.
00E4 C3 41 11 05FA 1923 BRB 66$ : Go finish logged message.
05FC 1924 35$: MOVZBL PPDS$B_PORT(R2),- : Get remote station address from
0602 1925 UCB$N_RSADDR(R3) : packet.
00E8 C3 51 B4 0602 1926 CLRW UCB$N_RSADDR+4(R3) : Zero extend it to 48 bits.
51 D5 0606 1927 TSTL R1 : Do we have a PB address?
50 30 A1 13 0608 1928 BEQL 50$ : Branch if no and none exists.
060A 1929 MOVL PB$S_SBLINK(R1),R0 : Get SB address from PB.
060E 1930 BEQL 50$ : Branch if no SB available
00EA C3 18 A0 D0 0610 1931 MOVL SB$B_SYSTEMID(R0),- : Copy system id from system block
0616 1932 UCB$N_RSID(R3) : to the log entry.
00EE C3 1C A0 B0 0616 1933 MOVW SB$B_SYSTEMID+4(R0),-
061C 1934 UCB$N_RSID+4(R3)
50 53 DD 061C 1935 50$: PUSHL R3 : Save UCB address.
50 08 A2 32 061E 1936 CVTWL PPDS$W_SIZE(R2),R0 : Get possible neg offset to net hdr
0A 18 0622 1937 BGEQ 55$ : Branch if no net header
50 08 A240 9E 0624 1938 MOVAB PPDS$W_SIZE(R2)[R0],R0 : Else get addr of net header
50 60 08 A2 A1 0629 1939 ADDW3 PPDS$W_SIZE(R2),(R0),R0 : and get size stored in net header
```



```

      55  50  0C  A3  062E 1940 55$: SUBW3 #PPDSB_PORT, R0, R5      ; - size of net header
      O0BC 8F  00  0C  A2  55  2C  062E 1941 55$:      ; Compute maximum length of message
      00F0 C3      0632 1942      ; based upon allocated pool region.
      063A      0632 1943      ; Move all interesting parts of the
      063D 1944      ; message packet to the logged
      063D 1945      ; message buffer.
      063D 1946 66$: POPL R3      ; Restore UCB address.
      51  50  03  9A  0640 1947      ; Get CI logged message sub-type code.
      0068 8F  3C  0643 1948      ; Get size of logged message.
      00D0 C3  B1  0648 1949      ; Is it a plain (short) logged msg?
      4007 8F      064C 1950      ; <PAERSK_ET_PKT28 + PAERSK_ES_ERRDGT>
      51  00F4 C3  12  064F 1951      ; Branch if so
      00F4 C3  3C  0651 1952      ; Get a copy of the PPD length from
      51  26  C0  0656 1953      ; the saved message
      00000DC 8F  51  D1  0659 1955      ; Add in other parts of error log entry
      07  15  0660 1957      ; Is it more than we will log?
      51  00000DC 8F  D0  0662 1958      ; Branch if not
      0669 1959      ; Else put in max errlog entry size
      7E  50  7D  0669 1960 80$: MOVQ R0, -(SP)      ; Save registers
      51  FA3A CF  9E  066C 1961      ; Retrieve logged message OPA0 table
      00008000 8F  CB  0671 1962      ; Retrieve error subtype and type and
      50  00D0 C3      0677 1963      ; clearing port crash indicating bit
      1D  10  067B 1964      ; Log the error to OPA0 if indicated
      53  55  00000A0 8F  C3  067D 1965      ; Compute UCB address
      50  8E  7D  0685 1966      ; Restore registers
      0688 1967      ;
      52  00D0 C3  9E  0688 1968      ; Get starting address of message.
      00000000 GF  16  068D 1969      ; Log the message.
      64 A3  04  AA  0693 1970      ; <ERL$LOGMESSAGE
      3F  BA  0697 1971 90$: POPR #UCBSM_ERLOGIP, UCBSW_ST$ (R3) ; Clear err. log in progress flag.
      05  0699 1972      ; Restore saved registers.
      RSB      ; Return to caller.
```

```
069A 1974 .SBTTL OPAO_LOG, _OPAO ERROR LOGGING ROUTINE
069A 1975
069A 1976 :+
069A 1977 : This routine first determines whether or not _OPAO error logging should be
069A 1978 : done. Then, if logging to _OPAO is indicated, this routine saves what optional
069A 1979 : formatting information will be needed and creates a fork process, using the
069A 1980 : port UCB's message fork block, to handle the formatting and broadcasting of
069A 1981 : the appropriate error log message. If this fork block is currently in use,
069A 1982 : presumably for the broadcasting of an earlier error log message, the
069A 1983 : assumption is made that this earlier message is the more important one, and
069A 1984 : the error condition currently being processed is not logged to _OPAO.
069A 1985
069A 1986 : Error logging to _OPAO will be attempted whenever the system device, which
069A 1987 : is assumed to be the same as the error logging device, is currently
069A 1988 : unavailable. Such error logging will also always be done for certain error
069A 1989 : conditions, such as fatal port initialization errors.
069A 1990
069A 1991 : Inputs:
069A 1992
069A 1993 : IPL -Device or Fork IPL
069A 1994 : R0 -High word 0, Error Subtype, Error Type
069A 1995 : R1 -Address of an _OPAO Error Logging Table
069A 1996 : R3 -Address of UCB
069A 1997
069A 1998 : It is assumed that the logged message buffer portion of the UCB has
069A 1999 : been initialized for all error conditions which use the logged
069A 2000 : message error log entry format. The contents of device registers
069A 2001 : are always obtained via the PDT.
069A 2002
069A 2003 : Outputs:
069A 2004
069A 2005 : R0-R1, R3-R4 -Destroyed
069A 2006 : R5 -Address of UCB message fork block
069A 2007 : Other registers -Preserved
069A 2008 :-
069A 2009
069A 2010 .ENABL LSB
069A 2011 OPAO_LOG:
069A 2012 MOVAB UCB$$_MSGFKBLK(R3),R5 ; Retrieve fork block address into R5
069F 2013
069F 2014 :
069F 2015 : Find the entry in the appropriate _OPAO error log table that corresponds
069F 2016 : to the error condition currently being processed.
069F 2017 :
069F 2018
069F 2019 10$: CMPW R0,(R1) ; Entry for current error condition?
06A2 2020 BEQL 20$ ; Branch if so
06A4 2021 TSTW (R1) ; Have we reached the end of the table?
06A6 2022 BLSS 30$ ; Don't perform logging if we have
06A8 2023 ADDL2 #OPAO_LOG_SIZE,R1 ; Else, position to next table entry
06AB 2024 BRB 10$ ; Continue search
```

55 00A0 C3 9E

61 50 B1 069F 2019 10\$: CMPW R0,(R1) ; Entry for current error condition?

09 13 06A2 2020 BEQL 20\$ ; Branch if so

61 B5 06A4 2021 TSTW (R1) ; Have we reached the end of the table?

2C 19 06A6 2022 BLSS 30\$ ; Don't perform logging if we have

51 08 C0 06A8 2023 ADDL2 #OPAO\_LOG\_SIZE,R1 ; Else, position to next table entry

F2 11 06AB 2024 BRB 10\$ ; Continue search

```
06AD 2026
06AD 2027
06AD 2028 : The current error condition will be logged to _OPA0 under the following
06AD 2029 : circumstances:
06AD 2030
06AD 2031 : 1. It is indicated that such error conditions are always to be logged.
06AD 2032 : 2. The system disk has not yet been mounted.
06AD 2033 : 3. The system disk is currently being mounted.
06AD 2034 : 4. The system disk is undergoing mount verification.
06AD 2035 : 5. During mount verification it is discovered that the system disk drive
06AD 2036 : contains the wrong volume.
06AD 2037 : 6. The system disk has timed out.
06AD 2038 : 7. The local system is participating in a cluster and quorum has been lost.
06AD 2039
06AD 2040 : An implicit assumption is that the system and error logging disk are one and
06AD 2041 : the same.
06AD 2042
06AD 2043
23 02 A1 00 E0 06AD 2044 20$: BBS #V_ALWAYS,CFLAGS(R1),40$: Go log if this error is always logged
06AD 2045
50 00000000'GF D0 06AD 2046 MOVL G^EXE$GL SYSUCB,R0 : Retrieve UCB for system disk
34 A0 D5 06AD 2047 TSTL UCBSL_VCB(R0) : Has the system disk been mounted?
17 13 06AD 2048 BEQL 40$ : Go log if it hasn't
B3 06AD 2049 BITW #UCBSM MOUNTING!- : Is system disk in one of these states?
06AD 2050 UCBSM_WRONGVOL!- : Currently being mounted?
06AD 2051 UCBSM_MNTVERIP!- : Wrong volume in device?
06AD 2052 UCBSM_TIMEOUT- : Mount verification in progress?
64 A0 C240 8F 06AD 2053 UCBSW_STS(R0) : Timed out?
OF 12 06AD 2054 BNEQ 40$ : Go log if it is
06AD 2055
50 00000000'GF D0 06AD 2056 MOVL G^CLUS$GL CLUB,R0 : Retrieve cluster block
05 13 06AD 2057 BEQL 30$ : No need to log if there isn't one
1C E1 06AD 2058 BBC #CLUB$V_QUORUM,- : Go log if the system is participating
01 1C A0 06AD 2059 CLUB$CL_FLAGS(R0),40$ : in a cluster which has lost quorum
05 06AD 2060 30$: RSB : Return
02 E2 06AD 2061 40$: BBSS #UCB V MSGFKLOCK,- : Indicate msg fork block now in use
FA 68 A3 06AD 2062 UCBSQ_DEVSTS(R3),30$ : If the fork block already in use,
06AD 2063 : assume prior error condition is more
06AD 2064 : important & skip logging of this one
```

```
06DA 2066
06DA 2067
06DA 2068
06DA 2069
06DA 2070
06DA 2071
06DA 2072
06DA 2073
06DA 2074
06DA 2075
06DA 2076
06DA 2077
06DA 2078
06DA 2079
06DA 2080
06DA 2081
06DA 2082
06DA 2083
06DA 2084
06DA 2085
06DA 2086
06DA 2087
06DA 2088
06DA 2089
06DA 2090
06DA 2091
06DA 2092
06DA 2093
06DA 2094
06DA 2095
06DA 2096
06DA 2097
06DA 2098
06DA 2099
06DA 2100
06DA 2101
06DA 2102
06DA 2103

: A decision has been made to log the error condition to _OPAO. First, store
: within the UCB any optional information which will be required to format the
: _OPAO error log message. Finally setup and create a fork process to format
: and broadcast the appropriate error log message to _OPAO. The fork process is
: created using the UCB's message fork block.

09 02 A1 02 E1 06DA 2075 BBC #V RPORT,CFLAGS(R1),50$ : Remote port number required?
00E4 C3 9A 06DF 2076 MOVZBL UCB$N_RSADDR(R3),- : If so, then save the remote port
00B8 C3 42 06E3 2077 UCB$_OPAO_TEMP(R3) : number in UCB, and go setup and
: create the fork process
09 02 A1 03 E1 06E8 2079 BRB 70$
00F0 C3 D0 06ED 2081 50$: BBC #V PKT,CFLAGS(R1),60$ : CI packet information required?
00B8 C3 34 06F1 2082 MOVL UCB$_CICMD(R3),- : If so, then save the CI packet
: information in the UCB, and go setup
: and create the fork process
2F 02 A1 04 E1 06F6 2084 BRB 70$
54 0084 C3 D0 06FB 2086 60$: BBC #V REGS,CFLAGS(R1),70$ : Branch if device regs not required
00B8 C3 7C 0700 2087 MOVL UCB$_PDT(R3),R4 : Retrieve PDT address
00C0 C3 D4 0704 2088 CLRQ UCB$_OPAO_TEMP(R3) : Clear UCB locations where the device
: registers will be saved
: Protect device register references
: from machine checks
00E4 D4 D0 0714 2091 MOVL @PDT$_CNF(R4),- : Store contents of configuration
: register
00B8 C3 D0 0718 2092 MOVL UCB$_OPAO_TEMP(R3) : Store contents of port maintenance
: control register
00E8 D4 D0 071B 2093 MOVL @PDT$_PMCT(R4),- : Store contents of port status register
00BC C3 D0 071F 2094 MOVL UCB$_OPAO_TEMP+4(R3)
00EC D4 D0 0722 2095 MOVL @PDT$_PS(R4),-
00C0 C3 D0 0726 2096 UCB$_OPAO_TEMP+8(R3)
0729 2097 $PRTCTEND 65$ : If check occurs, leave zero values(s)
072A 2098
072A 2099 70$: MOVL R1,R4 : Save table entry for error in R4
00000739'EF 9F 072D 2100 PUSHAB OPAO LOG FORK : Fork process routine address
00000000'GF 17 0733 2101 JMP G*EXE$FORK : Fork ...
0739 2102
0739 2103 .DSABL LSB
```



```
0739 2105 .SBTTL OPAO_LOG_FORK, OPAO ERROR LOGGING
0739 2106 .SBTTL - FORK PROCESS ROUTINE
0739 2107
0739 2108
0739 2109 :+ This is the routine which assumes control, within the context of a fork
0739 2110 : process, when an error log message is to be broadcast to _OPAO. This routine
0739 2111 : formats and broadcasts the _OPAO error log message as follows:
0739 2112 :
0739 2113 : 1. Optionally format the error log message utilizing information contained
0739 2114 : within the _OPAO error log table entry for this specific error condition.
0739 2115 : The address of the appropriate table entry maybe found within R4 on input
0739 2116 : to the routine.
0739 2117 : 2. Release the message fork block by clearing the interlock bit. This step
0739 2118 : must be delayed until after the optional formatting is completed because
0739 2119 : the optional formatting makes use of UCB locations which we can not allow
0739 2120 : to be overwritten until we are through with them.
0739 2121 : 3. Copy the device controller letter into the error log message.
0739 2122 : 4. Broadcast the _OPAO error log message.
0739 2123 : 5. Broadcast a second message indicating that the port will be taken offline
0739 2124 : if this is indicated for this error condition (Fatal port initialization
0739 2125 : errors only).
0739 2126 :
0739 2127 : Inputs:
0739 2128 :
0739 2129 : R3 -Address of UCB
0739 2130 : R4 -Address an _OPAO Error Logging Table Entry
0739 2131 : R5 -Address of Message Fork Block
0739 2132 :
0739 2133 : It is assumed that the three longwords beginning at UCB$_OPAO_TEMP
0739 2134 : have been initialized with whatever values will be required to complete
0739 2135 : any optional formatting of the current _OPAO error log message.
0739 2136 :
0739 2137 : Outputs:
0739 2138 :
0739 2139 : R0-R5 -Destroyed
0739 2140 : Other registers -Preserved
0739 2141 :-
0739 2142 :
0739 2143 : .ENABL LSB
0739 2144 OPAO_LOG_FORK:
0739 2145 MOVL UCB$_CRB(R3),R0 : Retrieve CRB address
0739 2146 PUSHL CRB$_INTD+- : Retrieve and save address of
0740 2147 VEC$_INITIAL(R0) : controller initialization routine
0740 2148
0740 2149 CVTWL MSG(R4),R2 : Retrieve offset to counted message
0744 2150 ADDL2 (SP),R2 : Compute address of counted message
0747 2151
0747 2152 MOVZWL FORMAT(R4),R0 : Retrieve offset to formatting routine
0748 2153 BEQL 10$, : Branch if no special formatting
074D 2154 ADDL2 (SP),R0 : Else compute formatting routine addr
0750 2155 JSB (R0) : Perform special formatting
0752 2156
0752 2157 10$: BICB2 #UCB_M_MSGFKLOCK,- : Mark message fork block as being
0754 2158 UCB$_DEVSTS(R3) : no longer in use
0756 2159
0756 2160 MOVZBL (R2)+,R1 : Retrieve size and address of message
0759 2161 CLRL (SP) : Assume will not broadcast "Offline"
```

```
03 02 A4 01 E1 0758 2162 BBC #V_OFFLINE,CFLAGS(R4),20$; Branch if this is true
      6E 01 D0 0760 2163 MOVL #1,(SP) ; Else this second msg will be broadcast
      54 28 A3 D0 0763 2164
55 00000000'GF 9E 0767 2166 20$: MOVAB UCB$DDB(R3),R4 ; Get DDB address into R4
      17 A4 90 076E 2167 ; Get _OPA0 UCB address into R5
      06 A2 16 0771 2168 MOVAB DDB$T_NAME+3(R4),- ; Copy device controller letter from
00000000'GF 16 0773 2170 CTRLR_NAME(R2) ; DDB to ASCII message
      8E D5 0779 2171 JSB G^IOC$BROADCAST ; Send message to terminal driver
      01 12 077B 2172 TSTL (SP)+ ; Should the 'Offline' msg be broadcast?
      05 077D 2173 BNEQ 30$ ; Go do so if it should
      077E 2174 RSB ; Else return
52 00000000'EF 9E 077E 2176 30$: MOVAB INISMSG OFFL,R2 ; Retrieve counted message address
      51 82 9A 0785 2177 MOVZBL (R2)+,RT ; Retrieve message size and address
      17 A4 90 0788 2178 MOVAB DDB$T_NAME+3(R4),- ; Copy device controller letter from
      06 A2 17 078B 2179 CTRLR_NAME(R2) ; DDB to ASCII message
00000000'GF 17 078D 2180 JMP G^IOC$BROADCAST ; Send message to terminal driver and
      0793 2181 ; return
      0793 2182 .DSABL LSB
```

```
0793 2184 .SBTTL _OPAO ERROR LOGGING FORMATTING ROUTINES
0793 2185 .SBTTL = ERR$CNV_HEX_DEC ROUTINE TO CONVERT A BINARY NUMBER
0793 2186 .SBTTL - INTO ITS DECIMAL ASCII EQUIVALENCE
0793 2187
0793 2188
0793 2189 :+ This routine takes a binary number, converts it into a decimal number, and
0793 2190 : then converts the decimal number into its ASCII equivalence. An implicit
0793 2191 : assumption is made that the binary number to be converted fits in a byte
0793 2192 : (ie - has a value in the range 0 - 255 decimal).
0793 2193
0793 2194 : Inputs:
0793 2195
0793 2196 R0 -Number to convert into its ASCII equivalence
0793 2197 R2 -Field in which to store the result
0793 2198
0793 2199 : Outputs:
0793 2200
0793 2201 R0-R1,R3 -Destroyed
0793 2202 Other registers -Preserved
0793 2203 :-
0793 2204
0793 2205 .ENABL LSB
0793 2206 ERR$CNV_HEX_DEC::
0793 2207 _MOVAB CONV_TABLE,R3 ; Retrieve address of conversion table
0793 2208 MOVW #^A/ /,(R2) ; Blank out first two bytes of field
0793 2209
0793 2210 CLRL R1 ; Clear high order longword
0793 2211 EDIV #100,R0,R1,R0 ; Determine number of 100s and remainder
0793 2212 BEQL 10$ ; Branch if no 100s
0793 2213 MOVAB (R3)[R1],(R2) ; Otherwise store number in 100s place
0793 2214
0793 2215 10$: CLRL R1 ; Determine number of 10s and remainder
0793 2216 EDIV #10,R0,R1,R0 ; Determine number of 10s and remainder
0793 2217 BEQL 20$ ; Branch if no 10s
0793 2218 MOVAB (R3)[R1],1(R2) ; Store number in 10s place
0793 2219
0793 2220 20$: MOVAB (R3)[R0],2(R2) ; store number in 1s place
0793 2221 RSB ; Return
0793 2222 .DSABL LSB
```

53 F869 CF 9E 0793 2207  
62 2020 8F 80 0798 2208  
51 D4 079D 2209  
00000064 8F 7B 079F 2211  
04 13 07AB 2212  
62 6341 90 07AA 2213  
07AE 2214  
51 D4 07AE 2215  
0A 7B 07B0 2216  
05 13 07B5 2217  
01 A2 6341 90 07B7 2218  
07BC 2219  
02 A2 6340 90 07BC 2220  
05 07C1 2221  
07C2 2222

```
07C2 2224      .SBTTL -      FORMAT_PKT,      ROUTINE TO FORMAT PACKET
07C2 2225      .SBTTL -      INFORMATION
07C2 2226
07C2 2227
07C2 2228      * This routine formats packet information fields within an _OPA0 error log
07C2 2229      message. The formatted packet field appears in the message as follows:
07C2 2230
07C2 2231      FLAGS/OPC/STATUS/PORT      xx/xx/xx/xx
07C2 2232
07C2 2233      The packet fields are formatted from left to right by calling the routine
07C2 2234      HEX_TO_ASCII for each packet field to be formatted.
07C2 2235
07C2 2236      Inputs:
07C2 2237
07C2 2238      R2      -Address of _OPA0 Error Log Message
07C2 2239      R3      -Address of the UCB
07C2 2240      R4      -Address of an _OPA0 Error Logging Table Entry
07C2 2241
07C2 2242      It is assumed that UCB$_OPA0_TEMP has been initialized with the packet
07C2 2243      information to be formatted.
07C2 2244
07C2 2245
07C2 2246      Outputs:
07C2 2247
07C2 2248      R0-R1      -Destroyed
07C2 2249      Other registers      -Preserved
07C2 2250
07C2 2251
07C2 2252      .ENABL      LSB
07C2 2253      FORMAT_PKT:
07C2 2254      PUSH      #^M<R2,R3,R4,R5,R6>      ; Save some registers
07C2 2255      CVTBL      OFFSET(R4),R0      ; Retrieve offset to field to format
07C2 2256      ADDL2      R0,R2      ; Compute address of field to format
07C2 2257      MOVAB      UCB$_OPA0_TEMP+4(R3),R5      ; Get addr of 1st byte past pkt fields
07C2 2258      MOVZBL      #4,R6      ; Num of packets fields to be formatted
07C2 2259
07C2 2260      10$:      MOVZBL      -(R5),R1      ; Get contents of next field to format
07C2 2261      MOVL      #2,R0      ; Set number of nibbles in packet field
07C2 2262      BSBB      HEX_TO_ASCII      ; Format the current packet field
07C2 2263      INCL      R2      ; Step over the delimiter
07C2 2264      SOBGTR      R6,10$      ; Continue until all fields formatted
07C2 2265
07C2 2266      POPR      #^M<R2,R3,R4,R5,R6>      ; Restore registers
07C2 2267      RSB
07C2 2268      .DSABL      LSB
```

50 007C 8F BB 07C2 2254  
03 A4 98 07C6 2255  
52 50 C0 07CA 2256  
55 00BC C3 9E 07CD 2257  
56 04 9A 07D2 2258  
51 75 9A 07D5 2259  
50 02 D0 07D8 2261  
6A 10 07DB 2262  
52 D6 07DD 2263  
F3 56 F5 07DF 2264  
007C 8F BA 07E2 2265  
05 07E6 2267  
07E7 2268



```
07E7 2270      .SBTTL -      FORMAT_PORT,      ROUTINE TO FORMAT A
07E7 2271      .SBTTL -      REMOTE PORT NUMBER
07E7 2272
07E7 2273
07E7 2274      :+ This routine formats a remote port number field within an _OPA0 error log
07E7 2275      : message. The remote port number appears as a decimal number after formatting.
07E7 2276
07E7 2277      : Inputs:
07E7 2278
07E7 2279      : R2 -Address of _OPA0 Error Log Message
07E7 2280      : R3 -Address of the UCB
07E7 2281      : R4 -Address of an _OPA0 Error Logging Table Entry
07E7 2282
07E7 2283      : It is assumed that UCB$_OPA0_TEMP has been initialized with the
07E7 2284      : remote port number to be formatted.
07E7 2285
07E7 2286      : Outputs:
07E7 2287
07E7 2288      : R0-R1 -Destroyed
07E7 2289      : Other registers -Preserved
07E7 2290
07E7 2291      :-
07E7 2292
07E7 2293      .ENABL  LSB
07E7 2294  FORMAT_PORT:
07E7 2295      MOVQ  R2,-(SP)      ; Save registers
07EA 2296      CVTBL  OFFSET(R4),R0      ; Retrieve offset to field to format
07EE 2297      ADDL2  R0,R2      ; Compute address of field to format
07F1 2298      MOVL  UCB$_OPA0_TEMP(R3),R0      ; Retrieve remote port number
07F6 2299      BSBB  ERR$CNV_HEX_DEC      ; Format the remote port number
07F8 2300      MOVQ  (SP)+,R2      ; Restore registers
07FB 2301      RSB      ; Return
07FC 2302      .DSABL  LSB
```

50 7E 52 7D 07E7 2295  
50 03 A4 98 07EA 2296  
52 50 C0 07EE 2297  
50 00B8 C3 D0 07F1 2298  
9B 10 07F6 2299  
52 8E 7D 07F8 2300  
05 07FB 2301  
07FC 2302

```
07FC 2304 .SBTTL - FORMAT_REGS, ROUTINE TO FORMAT PORT
07FC 2305 .SBTTL - REGISTERS
07FC 2306
07FC 2307
07FC 2308 :+ This routine formats the port register fields within an OPA0 error log
07FC 2309 : message. Only the contents of selected port registers are formatted. The
07FC 2310 : formatted register fields appear in the message as follows:
07FC 2311 :
07FC 2312 : CNF/PMC/PSR xxxxxxxx/xxxxxxx/xxxxxxx
07FC 2313 :
07FC 2314 : The port register fields are formatted from left to right by calling the
07FC 2315 : routine HEX_TO_ASCII for each register field to be formatted.
07FC 2316 :
07FC 2317 : Inputs:
07FC 2318 :
07FC 2319 : R2 -Address of OPA0 Error Log Message
07FC 2320 : R3 -Address of the UCB
07FC 2321 : R4 -Address of an OPA0 Error Logging Table Entry
07FC 2322 :
07FC 2323 : It is assumed that the three longwords beginning at UCB$T_OPA0_TEMP
07FC 2324 : have been initialized with the values of the device registers to be
07FC 2325 : formatted.
07FC 2326 :
07FC 2327 :
07FC 2328 : Outputs:
07FC 2329 :
07FC 2330 : R0-R1 -Destroyed
07FC 2331 : Other registers -Preserved
07FC 2332 :-
07FC 2333 :
07FC 2334 : .ENABL LSB
07FC 2335 :
07FC 2336 : FORMAT_REGS:
07FC 2337 : PUSH R2,R3,R4,R5,R6 ; Save some registers
07FC 2338 : CMTBL OFFSET(R4),R0 ; Retrieve offset to field to format
07FC 2339 : ADDL2 R0,R2 ; Compute address of field to format
07FC 2340 : MOVAB UCB$T_OPA0_TEMP(R3),R5 ; Get address of first port register
07FC 2341 : MOVZBL #3,R6 ; Num of register fields to be formatted
07FC 2342 :
07FC 2343 : 10$:
07FC 2344 : MOVL (R5)+,R1 ; Get contents of next port register
07FC 2345 : MOVL #8,R0 ; Set number of nibbles in packet field
07FC 2346 : BSBW HEX_TO_ASCII ; Format the current port register field
07FC 2347 : INCL R2 ; Step over the delimiter
07FC 2348 : SOBGTR R6,10$ ; Continue until all registers formatted
07FC 2349 :
07FC 2350 : POP R2,R3,R4,R5,R6 ; Restore registers
07FC 2351 : RSB ; Return
07FC 2352 : .DSABL LSB
```

50 007C 8F BB 07FC 2336  
03 A4 98 0800 2337  
52 50 C0 0804 2338  
55 00B8 C3 9E 0807 2339  
56 03 9A 080C 2340  
080F 2341  
51 85 D0 080F 2342  
50 08 D0 0812 2343  
002F 30 0815 2344  
52 D6 0818 2345  
F2 56 F5 081A 2346  
081D 2347  
007C 8F BA 081D 2348  
05 0821 2349  
0822 2350

```
0822 2352 .SBTTL - FORMAT_REV, FORMAT PORT UCODE REV LEVELS
0822 2353
0822 2354 :+
0822 2355 : This routine formats the PROM and RAM revision levels within an OPA0 message.
0822 2356 : The formatted field appears in the message as follows:
0822 2357 :
0822 2358 :     RAM/PROM rev is xxxx/xxxx
0822 2359 :
0822 2360 : The fields are formatted from left to right by calling the routine
0822 2361 : HEX_TO_ASCII for each rev.
0822 2362 :
0822 2363 : Inputs:
0822 2364 :
0822 2365 :     R2 -Address of OPA0 error message
0822 2366 :     R3 -Addr of UCB
0822 2367 :     R4 -Addr of OPA0 error message table entry
0822 2368 :
0822 2369 : It is assumed that UCB$T_OPA0_TEMP has been initialized with
0822 2370 : the rev level information to be formatted.
0822 2371 :
0822 2372 : Outputs:
0822 2373 :
0822 2374 :     R0,R1 -Destroyed
0822 2375 :     Other registers -Preserved
0822 2376 : -
0822 2377 :
0822 2378 : .ENABL LSB
0822 2379
0822 2380 FORMAT_REV:
0822 2381
0822 2382 PUSH R2,R3,R4,R5,R6 ; Save caller's registers
0822 2383 CMTBL OFFSET(R4),R0 ; Retrieve offset to field to fmt
0822 2384 ADDL2 R0,R2 ; Compute addr of field to fmt
0822 2385 MOVAV UCB$T_OPA0_TEMP(R3),R5 ; Get addr of RAM rev
0822 2386 MOVZBL #2,R6 ; Two rev levels to fmt
0822 2387
0822 2388 10$: MOVW (R5)+,R1 ; Get next rev level
0822 2389 MOVL #4,R0 ; 4 hex digits/rev level
0822 2390 BSBB HEX_TO_ASCII ; Format this rev
0822 2391 INCL R2 ; Step past slash delimiter, /
0822 2392 SOBGTR R6,10$ ; Continue formatting revs
0822 2393 POPR #R2,R3,R4,R5,R6 ; Restore registers
0822 2394 RSB ; Return to caller
0822 2395
0822 2396 .DSABL LSB
```

50 007C 8F BB 0822 2382  
03 A4 98 0826 2383  
52 50 C0 082A 2384  
55 00B8 C3 3E 082D 2385  
56 02 9A 0832 2386  
51 85 B0 0835 2387  
50 04 D0 0838 2388  
0A 10 083B 2389  
52 D6 083D 2390  
F3 56 F5 083F 2391  
007C 8F BA 0842 2392  
05 0846 2393  
0847 2394  
0847 2395  
0847 2396

```

0847 2398      .SBTTL -      HEX_TO_ASCII  ROUTINE TO CONVERT A BINARY NUMBER
0847 2399      .SBTTL -      INTO ITS ASCII EQUIVALENCE
0847 2400
0847 2401      :+
0847 2402      : This routine takes a binary number, converts it into its ASCII equivalence,
0847 2403      : and stores it in the field provided. The nibbles of the binary number are
0847 2404      : processed and stored in their ASCII equivalences from left to right. This
0847 2405      : routine is capable of handling up to a longword at a time in this fashion.
0847 2406
0847 2407      Inputs:
0847 2408
0847 2409      R0      -Number of nibbles in field to be converted
0847 2410      R1      -Number to convert into its ASCII equivalence
0847 2411      R2      -Field in which to store the ASCII equivalences
0847 2412
0847 2413      Outputs:
0847 2414
0847 2415      R0,R3-R4  -Destroyed
0847 2416      R2      -Address of first byte past field
0847 2417      Other registers  -Preserved
0847 2418      :-
0847 2419
0847 2420      .ENABL  LSB
0847 2421  HEX_TO_ASCII:
0847 2422      MOVAB  CONV_TABLE,R3      ; Retrieve address of conversion table
0847 2423      ASHL   #2,R0,R0          ; Compute bit number of leftmost nibble
0850 2424      SUBL2  #4,R0          ; which is to be converted
0853 2425
0853 2426  10$:  EXTZV  R0,#4,R1,R4      ; Extract the current nibble
0858 2427      MOVB  (R3)[R4],(R2)+      ; Move ASCII equivalence into field
085C 2428      ACBB  #0,#-4,R0,10$      ; Continue until all nibbles processed
0863 2429      RSB
0864 2430      .DSABL  LSB
0864 2431
0864 2432      .END

```

```

53  F7B5 CF  9E
50  50  02  78
   50  04  C2
54  51  04  50  EF
   82  6344  90
FFFO 50  FC 8F  00  9D
   05

```



PAERROR  
Symbol table

Error Handling & Logging Routines M 16

16-SEP-1984 01:16:25 VAX/VMS Macro V04-00  
10-SEP-1984 01:16:10 [DRIVER.SRC]PAERROR.MAR;2

Page 54  
(35)

```

$$$CURSZ      = 000001C4
$$$NEWSIZ     = 000001D0
$$MSG_PTR     = 00000954 R      03
BELL          = 00000007
BUGS_CIPORT   = ***** X      01
CDTSC_CON_REC = 00000009
CDTSC_VC_FAIL = 0000000C
CDTSL_CDTLST  = 0000006C
CDTSM_STATE   = 00000028
CFLAGS       = 00000002
CLN_BYTES    = 00000014
CLUGL_CLUB   = ***** X      01
CLURSL_FLAGS = 0000001C
CLUBSV_QUORUM = 0000001C
CNFSLKP_PB_MSG = ***** X      01
CNFSLKP_PB_PDT = ***** X      01
CNFSREMOVE_PB = ***** X      01
COMSDRVDEACMEM = ***** X      01
CONV_TABLE    = 00000000 R      01
CR            = 0000000D
CRBSL_INTD    = 00000024
CTRLR_NAME    = 00000006
DA_MASK       = 0000003E
DA_OPAO_LOG_TAB = 00000010 R      01
DDBST_NAME    = 00000014
ELOG$$LOG_DA  = 0000043B R      01
ELOG$$LOG_LM  = 00000526 R      01
ELOG$CABLES   = 000004F3 RG     01
ELOG$CBL_X_CHG = 000004E4 RG     01
ELOG$CPU_REV  = 000003F3 RG     01
ELOG$ERROR_DG = 00000518 RG     01
ELOG$HARDWARE = 0000041A RG     01
ELOG$INIT_SWERR = 0000038D RG     01
ELOG$INTRLOCK = 00000425 RG     01
ELOG$K_BYTES  = 0000007A G
ELOG$PACKET   = 00000504 RG     01
ELOG$PACKET1  = 0000050D RG     01
ELOG$PTH_ST_CHG = 000004CE RG     01
ELOG$REGDUMP  = 00000476 RG     01
ELOG$UCODE_ERR = 000003FE RG     01
ELOG$UCODE_NORD = 000003C9 RG     01
ELOG$UCODE_WARN = 00000407 RG     01
EMBSL_PM      = 00000003
EMBSL_DV_REGS = 0000004E
ERL$DEV:TEATTN = ***** X      01
ERL$LOGMESSAGE = ***** X      01
ERR$BUGCHECK  = 0000039D RG     01
ERR$BUGCHECKNF = 0000037D RG     01
ERR$CLEANUP_PKT = 000002DC RG     01
ERR$CNV_HEX_DEC = 00000793 RG     01
ERR$CRASHVC   = 00000154 RG     01
ERR$CRASH_PORT = 0000018D RG     01
ERR$DEBUGCHECK = 000003B9 RG     01
ERR$DISC_PWF_FAIL = 000002A6 RG     01
ERR$DISP_ENTRY = 00000332 RG     01
ERR$INIPORT   = 00000355 RG     01
ERR$PWF_RECOV = 000001C1 RG     01

```

```

ERRSV_DEB_ABO      = 00000008 G
ERRSV_DEB_ACCV     = 00000002 G
ERRSV_DEB_BLV      = 00000001 G
ERRSV_DEB_BUGNF    = 00000014 G
ERRSV_DEB_CNFR     = 00000011 G
ERRSV_DEB_ILKQ     = 00000012 G
ERRSV_DEB_INVBN    = 00000000 G
ERRSV_DEB_INVDP    = 00000003 G
ERRSV_DEB_INVOP    = 00000008 G
ERRSV_DEB_MFQE     = 00000018 G
ERRSV_DEB_NEPQ     = 00000013 G
ERRSV_DEB_NOPB     = 00000010 G
ERRSV_DEB_NOSTS    = 0000000D G
ERRSV_DFB_NPUPD    = 00000009 G
ERRSV_DEB_OSEQ     = 00000016 G
ERRSV_DEB_PSRX     = 00000015 G
ERRSV_DEB_PSV      = 00000003 G
ERRSV_DEB_SCERR    = 0000000F G
ERRSV_DEB_UNSTS    = 0000000C G
ERRSV_DEB_URC      = 00000007 G
ERRSV_DEB_URP      = 00000005 G
ERRSV_DEB_VCDCL    = 00000017 G
ERRSV_DEB_VCUPD    = 0000000A G
ERRSV_DEB_XCTER    = 0000000E G
EXES$FORK         = ***** X      01
EXES$GL_DEFFLAGS  = ***** X      01
EXES$GL_LOCKRTRY  = ***** X      01
EXES$GL_SYSUCB    = ***** X      01
EXES$MCRK_PRTCT  = ***** X      01
EXESV_FATAL_BUG  = ***** X      01
FATALQ           = 00000349 R      01
FLUSH_Q          = 00000319 R      01
FORMAT           = 00000004
FORMAT_PKT       = 000007C2 R      01
FORMAT_PORT      = 000007E7 R      01
FORMAT_REGS      = 000007FC R      01
FORMAT_REV       = 00000822 R      01
HEX_TO_ASCII     = 00000847 R      01
IDBSL_CSR        = 00000000
INIS$FORK        = ***** X      01
INISMSG_OFFL     = ***** X      01
INISPORT         = ***** X      01
INT$DEAL_PKT     = ***** X      01
INT$DISP_SENDDG  = ***** X      01
INT$INS_COMQH    = ***** X      01
IOC$BROADCAST    = ***** X      01
LF               = 0000000A
LM_MASK          = 0000003F
LM_OPAO_LOG_TAB  = 000000AA R      01
LOG_AS_CHANGE    = 000004ED R      01
LOG_AS_HARDWARE  = 00000420 R      01
MCHRSM_NEXM     = 00000004
MSG              = 00000006
M_ALWAYS         = 00000001
M_OFFLINE        = 00000002
M_PKT           = 00000008
M_REGS           = 00000010

```

PAERROR  
Symbol table

Error Handling & Logging Routines

B 1

16-SEP-1984 01:16:25 VAX/VMS Macro V04-00  
10-SEP-1984 01:16:10 [DRIVER.SRC]PAERRJR.MAR;2

Page 55  
(35)

M RPORT  
NUM\_EX\_LONGWORDS  
OFFSET  
OPASUCBO  
OPAO\_LOG  
OPAO\_LOG\_FORK  
OPAO\_LOG\_SIZE  
PASCTLINIT  
PAERSK\_ES\_OBG  
PAERSK\_ES\_OGB  
PAERSK\_ES\_1BG  
PAERSK\_ES\_1GB  
PAERSK\_ES\_CNPB  
PAERSK\_ES\_CODE  
PAERSK\_ES\_CPUREV  
PAERSK\_ES\_CSHP  
PAERSK\_ES\_CU  
PAERSK\_ES\_DOIN  
PAERSK\_ES\_DORM  
PAERSK\_ES\_ERRDG  
PAERSK\_ES\_HCIN  
PAERSK\_ES\_HWER  
PAERSK\_ES\_INIT  
PAERSK\_ES\_LOBG  
PAERSK\_ES\_LOBX  
PAERSK\_ES\_LOGB  
PAERSK\_ES\_L1BG  
PAERSK\_ES\_L1BX  
PAERSK\_ES\_L1GB  
PAERSK\_ES\_LCIN  
PAERSK\_ES\_LST0  
PAERSK\_ES\_LST1  
PAERSK\_ES\_LST2  
PAERSK\_ES\_LST3  
PAERSK\_ES\_LST4  
PAERSK\_ES\_MQIN  
PAERSK\_ES\_MQRM  
PAERSK\_ES\_NOPB  
PAERSK\_ES\_PCVC  
PAERSK\_ES\_PDWN  
PAERSK\_ES\_POOL  
PAERSK\_ES\_PUP  
PAERSK\_ES\_REVCA  
PAERSK\_ES\_REVER  
PAERSK\_ES\_RQRM  
PAERSK\_ES\_RSCKS  
PAERSK\_ES\_SCA  
PAERSK\_ES\_SCSID  
PAERSK\_ES\_SCVC  
PAERSK\_ES\_UC  
PAERSK\_ES\_UCDW  
PAERSK\_ES\_UPKT  
PAERSK\_ES\_UXIN  
PAERSK\_ET\_CBL  
PAERSK\_ET\_DALT  
PAERSK\_ET\_HW  
PAERSK\_ET\_ILCK

= 00000004  
= 00000003  
= 00000003  
\*\*\*\*\*  
0000069A  
00000739  
= 00000008  
\*\*\*\*\*  
= 00000002  
= 00000000  
= 00000003  
= 00000001  
= 00000004  
= 00000001  
= 00000007  
= 00000002  
= 00000005  
= 00000006  
= 00000001  
= 00000007  
= 00000003  
= 00000002  
= 00000001  
= 00000008  
= 00000004  
= 00000006  
= 00000009  
= 00000008  
= 00000007  
= 00000004  
= 00000003  
= 00000009  
= 00000007  
= 00000009  
= 0000000C  
= 00000005  
= 00000000  
= 00000006  
= 00000001  
= 00000003  
= 00000000  
= 00000004  
= 00000008  
= 00000006  
= 00000002  
= 00000008  
= 00000005  
= 00000002  
= 00000003  
= 00000004  
= 00000000  
= 00000000  
= 00000005  
= 000000041  
= 00000003  
= 00000001  
= 00000002

X 01  
R 01  
R 01  
X 01

PAERSK\_ET\_INSW  
PAERSK\_ET\_LMLT  
PAERSK\_ET\_PKT  
PAERSM\_CPRT  
PA\_CNF  
PA\_CQ0  
PA\_CQ1  
PA\_CQ2  
PA\_CQ3  
PA\_C\_WCSSIZ  
PA\_DFQ  
PA\_MADR  
PA\_MDATR  
PA\_MFQ  
PA\_MTC  
PA\_MTEC  
PA\_PDC  
PA\_PEC  
PA\_PESR  
PA\_PEAR  
PA\_PIC  
PA\_PMC  
PA\_PMC\_M\_MIN  
PA\_PPR  
PA\_PQBBR  
PA\_PS  
PA\_PSR  
PB\$B\_PO\_STS  
PB\$B\_P1\_STS  
PB\$B\_RSTATION  
PB\$C\_LENGTH  
PB\$C\_PALENGTH  
PB\$C\_PWR\_FAIL  
PB\$C\_VC\_FAIL  
PB\$S\_CDTLST  
PB\$S\_CLSCKT\_DG  
PB\$S\_SBLINK  
PB\$M\_CUR\_CBL  
PB\$W\_STATE  
PDT\$B\_DQIMAP  
PDT\$B\_HSHUT\_DG  
PDT\$B\_MAX\_PORT  
PDT\$B\_NXT\_PORT  
PDT\$B\_PO\_CBSTS  
PDT\$B\_P1\_LBSTS  
PDT\$B\_PLOGMAP  
PDT\$B\_PORTMAP  
PDT\$B\_PORT\_NUM  
PDT\$B\_REQIDPS  
PDT\$C\_LENGTH  
PDT\$C\_PALENGTH  
PDT\$C\_PAREGBASE  
PDT\$C\_PAREGEND  
PDT\$C\_PQB  
PDT\$C\_CNF  
PDT\$C\_CQ0  
PDT\$C\_CQ1

= 00000000  
= 00000042  
= 00000040  
= 00000080  
00000000  
00000908  
0000090C  
00000910  
00000914  
= 00000C00  
00000928  
00000014  
00000018  
0000092C  
00000930  
00000934  
00000920  
0000091C  
0000093C  
00000938  
00000924  
00000004  
= 00000001  
00000940  
00000904  
00000900  
00000918  
= 00000029  
= 0000002A  
= 0000000C  
= 00000054  
00000060  
= 00004000  
= 00008000  
= 00000034  
00000054  
= 00000030  
= 00000001  
= 00000012  
00000154  
00000180  
0000017C  
0000017E  
00000180  
00000181  
00000134  
00000114  
0000017D  
0000017F  
= 000000E4  
= 00000360  
000000E4  
00000110  
= 000001E0  
000000E4  
000000F0  
000000F4

PAERROR  
Symbol table

Error Handling & Logging Routines

C 1

16-SEP-1984 01:16:25 VAX/VMS Macro V04-00  
10-SEP-1984 01:16:10 [DRIVER.SRC]PAERROR.MAR;2

Page 56  
(35)

PDTSL_DFQ	000000FC
PDTSL_DFQHDR	00000208
PDTSL_DGHDRSZ	00000190
PDTSL_DGNETHD	00000194
PDTSL_DQEOLOGOUT	000002E0
PDTSL_GPTBASE	0000022C
PDTSL_GPTLEN	00000230
PDTSL_LBDG	00000184
PDTSL_MFQ	00000100
PDTSL_MFQHDR	0000020C
PDTSL_MQEOLOGOUT	00000320
PDTSL_MSGHDRSZ	= 000000B4
PDTSL_MTC	00000104
PDTSL_PPAR	00000108
PDTSL_PMC	000000E8
PDTSL_POLLERDUE	0000018C
PDTSL_POOLDUE	00000188
PDTSL_PPR	0000010C
PDTSL_PS	000000EC
PDTSL_PSR	000000F8
PDTSL_SPTBASE	00000224
PDTSL_SPTLEN	00000228
PDTSL_UCBO	= 000000DC
PDTSL_VBDT	0000021C
PDTSL_VPQB	00000218
PDTSM_PWF_CLNUP	= 00000001
PDTSQ_COMQ2	000001F0
PDTSQ_COMQ3	000001F8
PDTSQ_COMQBASE	000001E0
PDTSQ_COMQH	000001E8
PDTSQ_COMQL	000001E0
PDTSQ_DFREQ	000001D0
PDTSQ_FORMPB	00000174
PDTSQ_MFREQ	000001D8
PDTSQ_RSPQ	00000200
PDTSQ_TEMP_RSPQ	0000019C
PDTSV_PUP	= 00000001
PDTSV_PWF_CLNUP	= 00000000
PDTSW_BDTLEN	00000220
PDTSW_DQELN	00000210
PDTSW_LPORT_STS	00000110
PDTSW_MQELN	00000214
PDTSW_PBCOUNT	00000112
PDTSW_STGDYN	00000198
PDTSW_STDGUSED	0000019A
PORT_REGS_LOGGED	= 00000006
PORT_UCODE	0000040C R 01
PPDSB_DEF_ST	0000001C
PPDSB_FLAGS	0000000F
PPDSB_HUVERS	00000034
PPDSB_LBDATA	00000012
PPDSB_LCB_O	00000012
PPDSB_LCB_LPORT	00000010
PPDSB_LCB_NPORT	0000000F
PPDSB_LCB_OPC	00000011
PPDSB_LCB_PORT	0000000E
PPDSB_OPC	0000000E

PPDSB_PORT	0000000C
PPDSB_PROTOCOL	0000001A
PPDSB_RSTATE	00000025
PPDSB_RST_PORT	00000024
PPDSB_STATUS	0000000D
PPDSB_SWFLAG	0000000B
PPDSB_SYSTEMID	00000014
PPDSB_TYPE	0000000A
PPDSC_LB_LENGTH	00000046
PPDSC_LCB_DATA	00000013
PPDSC_LENGTH	00000012
PPDSC_MIN_DGSIZ	00000050
PPDSC_SECTKT	= 00000019
PPDSC_SNDDG	= 00000001
PPDSK_LB_LENGTH	00000046
PPDSK_LENGTH	00000012
PPDSL_BLINK	00000004
PPDSL_DG_DISC	00000028
PPDSL_FLINK	00000000
PPDSL_IN_VCD	00000018
PPDSL_LB_CRC	00000042
PPDSL_PO_ACK	00000010
PPDSL_PO_NAK	00000014
PPDSL_PO_NRSP	00000018
PPDSL_P1_ACK	0000001C
PPDSL_P1_NAK	00000020
PPDSL_P1_NRSP	00000024
PPDSL_REC_BOFF	00000028
PPDSL_REC_NAME	00000024
PPDSL_RPORT_FCN	00000020
PPDSL_RPORT_REV	0000001C
PPDSL_RPORT_TYP	00000018
PPDSL_SND_BOFF	00000020
PPDSL_SND_NAME	0000001C
PPDSL_ST_ADDR	00000018
PPDSL_XCT_LEN	00000018
PPDSM_CST	= 00008000
PPDSM_DISPOSE	= 00000001
PPDSM_RSP	= 00000001
PPDSQ_CURTIME	00000048
PPDSQ_NODENAME	00000040
PPDSQ_SWINCARN	00000028
PPDSQ_XCT_ID	00000010
PPDST_HWTYPE	00000030
PPDST_SWTYPE	00000020
PPDST_SWVERS	00000024
PPDSV_RSP	= 00000000
PPDSW_LCB_LEN7	0000000C
PPDSW_LENGTH	00000010
PPDSW_MASK	00000010
PPDSW_MAXDG	0000001C
PPDSW_MAXMSG	0000001E
PPDSW_MTYPE	00000012
PPDSW_M_VAL	00000014
PPDSW_SIZE	00000008
PR\$ IPL	***** X 01
Q_UNLOCKED	000002A5 R 01



PAERROR  
Symbol table

Error Handling & Logging Routines

D 1

16-SEP-1984 01:16:25 VAX/VMS Macro V04-00  
10-SEP-1984 01:16:10 [DRIVER.SRC]PAERROR.MAR;2

Page 57  
(35)

REV ERROR	= 00000414	R	01	UCBSW_DEVSTS	= 00000068	
SAVEEDRS	= 00000014			UCBSW_ERRCNT	= 00000082	
SB\$B_SYSTEMID	= 00000018			UCBSW_LMERRCNT	= 000000D4	
SB\$S_NODENAME	= 00000010			UCBSW_MSGBYTCNT	= 000000F4	
SB\$S_SYSTEMID	= 00000006			UCBSW_MSGPPDTYP	= 000000F6	
SB\$T_NODENAME	= 00000044			UCBSW_STS	= 00000064	
SCS\$CLOSE_CDT	*****	X	01	UCB_M_MSGFKLOCK	= 00000004	
SCS\$DEALL_CDT	*****	X	01	UCB_V_MSGFKLOCK	= 00000002	
SCS\$DEAL_SCSREC	*****	X	01	UNLOCK_BADQ	= 00000295	R 01
SCS\$FREE_LISTEN	*****	X	01	VECSL_IDB	= 00000008	
SCS\$GB_SYSTEMID	*****	X	01	VECSL_INITIAL	= 0000000C	
SCS\$GL_MCADR	*****	X	01	V_ALWAYS	= 00000000	
SCS\$NOTIFY_SYSAP	*****	X	01	V_OFFLINE	= 00000001	
SIZ...	= 00000001			V_PKT	= 00000003	
SS\$ABORT	= 0000002C			V_REGS	= 00000004	
SS\$CTRLERR	= 00000054			V_RPORT	= 00000002	
SS\$NORMAL	= 00000001					
SS\$POWERFAIL	= 00000364					
SUBTYPE	= 00000000					
TOTAL_LONGWORDS	= 0000000B					
TYPE	= 00000001					
UCBSB_DIPL	= 0000005E					
UCBSB_ERTCNT	= 00000080					
UCBSB_LMERTCNT	= 000000D2					
UCBSB_LMERTMAX	= 000000D3					
UCBSB_LMEST	= 000000D0					
UCBSB_LMET	= 000000D1					
UCBSK_ERRDGBYTS	= 000000B4					
UCBSK_ERRDGSIZ	= 000000DC					
UCBSK_LMBUFSIZ	= 00000068					
UCBSK_LMPKTBYTS	= 00000040					
UCBSL_CICMD	= 000000F0					
UCBSL_CRB	= 00000024					
UCBSL_DDB	= 00000028					
UCBSL_DPC	= 0000009C					
UCBSL_FR4	= 00000014					
UCBSL_MSGFKBLK	= 000000A0					
UCBSL_PDT	= 00000084					
UCBSL_VCB	= 00000034					
UCBSM_ERLOGIP	= 00000004					
UCBSM_MNTVERIP	= 00004000					
UCBSM_MOUNTING	= 00000200					
UCBSM_ONLINE	= 00000010					
UCBSM_TIMEOUT	= 00000040					
UCBSM_WRONGVOL	= 00008000					
UCBSN_LSADDR	= 000000D8					
UCBSN_LSID	= 000000DE					
UCBSN_RSADDR	= 000000E4					
UCBSN_RSID	= 000000EA					
UCB\$S_LSADDR	= 00000006					
UCB\$S_LSID	= 00000006					
UCB\$S_RSADDR	= 00000006					
UCB\$S_RSID	= 00000006					
UCB\$T_MSGDATA	= 000000F8					
UCB\$T_OPAO_TEMP	= 000000B8					
UCB\$V_ERLOGIP	= 00000002					
UCB\$V_ONLINE	= 00000004					



+-----+  
! Psect synopsis !  
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$\$\$115_DRIVER	00000864 ( 2148.)	01 ( 1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
\$AB\$\$	00000944 ( 2372.)	02 ( 2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$110_MSGS	000009A6 ( 2470.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+  
! Performance indicators !  
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.02	00:00:02.32
Command processing	133	00:00:00.46	00:00:03.31
Pass 1	624	00:00:19.74	00:01:11.93
Symbol table sort	0	00:00:02.31	00:00:09.17
Pass 2	412	00:00:05.30	00:00:18.81
Symbol table output	3	00:00:00.24	00:00:01.58
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1211	00:00:28.09	00:01:47.14

The working set limit was 2400 pages.

167702 bytes (328 pages) of virtual memory were used to buffer the intermediate code.

There were 120 pages of symbol table space allocated to hold 2150 non-local and 70 local symbols.

2432 source lines were read in Pass 1, producing 33 object records in Pass 2.

50 pages of virtual memory were used to define 47 macros.

+-----+  
! Macro library statistics !  
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[DRIVER.OBJ]PALIB.MLB;1	9
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	24
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	41

2482 GETS were required to define 41 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:PAERROR/OBJ=OBJ\$:PAERROR MSRC\$:PAERROR/UPDATE=(ENH\$:PAERROR)+EXECMLS/LIB+LIB\$:PALIB.MLB/LIB



0113 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

PA CONFIG  
LIS

PAEND  
LIS

PAERROR  
LIS



0114 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

PAINT  
LIS

PAINT  
LIS

PAINT  
LIS

PAINT  
LIS